Sveučilište u Zagrebu
Fakultet Strojarstva i Brodogradnje

DIPLOMSKI RAD

Igor Ratković

Zagreb, 2015.

Sveučilište u Zagrebu
Fakultet Strojarstva i Brodogradnje

DIPLOMSKI RAD

Mentor :

prof.dr.sc. Davor Zorc

Student:

Igor Ratković

Zagreb, 2015.

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite

Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

DIPLOMSKI ZADATAK

Student: **Igor Ratković** Mat. br.: 0035177506

Naslov rada na hrvatskom jeziku: UPRAVLJANJE I NADZOR REGULACIJSKIH OBJEKATA POMOĆU LOKALNOG I NADREĐENOG MIKROKONTROLERA

Naslov rada na engleskom jeziku: Control and supervision of control objects by means of local and superimposed microcontrollers

Opis zadatka:

Treba projektirati i izraditi edukativnu maketu iz područja mikroprocesorskog upravljanja i automatske regulacije. Dijelovi makete su: sekvencijalno upravljanje semafora, programabilni logički kontroler i automatska regulacija toplinskog procesa. Pritom je lokalni mikrokontroler Arduino Uno a nadređeni sustav je Raspberry Pi Linux računalo. Za razmatrane mikrokontrolerske sustave upravljanja treba razviti programsku podršku nadređenog sustava za simulaciju rada stvarnih objekata regulacije te za komunikaciju preko serijske veze, te upravljanje i grafički prikaz trenutnog stanja stvarnih objekata u radu. Na strani lokalog mikrokontrolera treba razviti upravljačke programe za sva tri objekta regulacije. Lokalni sustav treba biti primjenjiv i sa PC nadređenim sustavom. Kod regulacije toplinskog procesa treba analizirati svojstva toplinskog procesa te prednosti PID regulatora u odnosu na on-off regulaciju uz grafičku usporedbu. Napisati uputu za studente koji će koristiti ovaj sustav na laboratorijskim vježbama.

Zadatak zadan: 15. siječnja 2015.

Rok predaje rada: 19. ožujka 2015.

Predviđeni datum obrane: 25., 26. i 27. ožujka 2015.

Zadatak zadao: Prof. dr. sc. Davor Zorc

Predsjednik Povjerenstva: Prof. dr. sc. Franjo Cajner

Izjavljujem da sam ovaj radi izradio samostalno, služeći se znanjem stečenim tijekom studija, te navedenom literaturom i izvorima.

Zahvaljujem se mentoru prof. dr. sc. Davoru Zorcu na savjetima i potpori tijekom izrade ovog rada.
Također se zahvaljujem doc. dr. sc. Danijelu Pavkoviću na pruženoj pomoći i korisnim savjetima iz područja automatske regulacije i elektroničkih komponenata i sklopova.
Želim se zahvaliti mojoj obitelji na bezrezervnoj podršci i razumijevanju.
I posebno hvala mojim kolegama i prijateljima s fakulteta koji su mi uljepšali i olakšali studentski život.

Sadržaj

Popis tablica

Popis slika

Popis primjera programskog koda

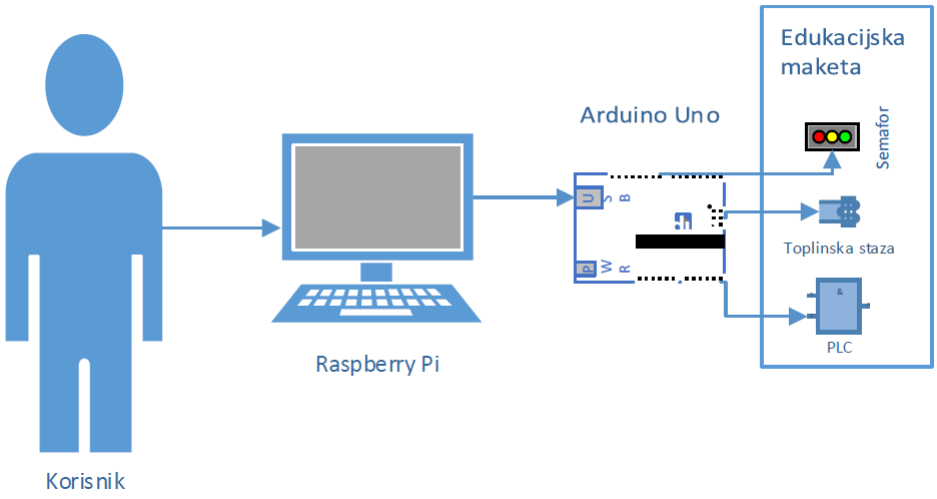
Sažetak

U radu je prikazan postupak modernizacije postojećih nastavnih maketa iz mikroprocesorskog upravljanja, elektronike i regulacije, zasnovan na primjeni otvorenog softvera i razmjerno pristupačnog upravljačkog hardvera pogodnog za edukacijske primjene, kao što su Arduino Uno mikrokontroler i Raspberry Pi kompaktno mikroračunalo. Pritom Arduino Uno mikrokontroler služi za izravno upravljanje pojedinim dijelovima nastavne makete na nižem nivou (engl. *slave*),a koje čine maketa semafora, emulator programabilnog logičkog kontrolera, toplinska komora, dok Raspberry Pi računalo služi kao terminal i nadređeno računalo (engl. *master*), pomoću kojeg se zadaju postavne veličine i mijenjaju parametri izvršnog koda na '*slave*' mikroračunalu. Predložena softverska rješenja temelje se na Linux operativnom sustavu, te C++ i Python razvojnim programskim okruženjima. Konačne izvedbe upravljačkih programa ispitane su najprije u virtualnom okruženju, te potom i na realnom objektu, odnosno nastavnoj maketi.

Ključne riječi: modernizacija, nastavna maketa, digitalni PID algoritam, programabilni logički kontroler, sekvencijalnu upravljanje, Arduino Uno, Raspberry Pi, C++, Python razvojno okruženje

1 Uvod

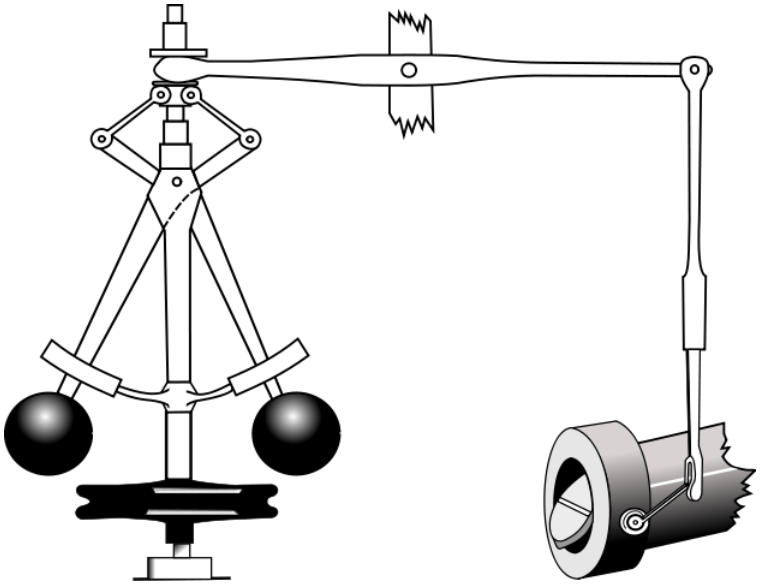
U ovom radu opisuje se postupak modernizacije edukacijske makete iz područja mikroprocesorskog upravljanja, te elektronike i automatske[1] regulacije. Modernizacija je provedena na dvije razine, na upravljačkoj razini mikrokontrolera i na nadređenoj razini u obliku korisničkog sučelja. Arduino Uno mikrokontroler spaja se na postojeće objekte regulacije edukacijske makete, i sadrži odgovarajuće programe za njihovo upravljanje. Arduino Uno [2] djeluje kao podređeni (engl. *slave*) mikrokontroler i putem serijske veze spaja se s nadređenim (engl. *master*) kontrolerom, Raspberry Pi[3] mikroračunalom. Modernizacija na razini grafičkog sučelja provedena je zamjenom konzolnog programa s intuitivnim grafičkim sučeljem koji korisniku omogućava simulaciju rada objekata regulacije edukacijske makete, slanje željenih parametara upravljanja putem serijske veze na podređeni Arduino mikrokontroler i praćenje rada programa. Unutar rada obrađena je problematika razvoja programa za upravljanje pojedinim dijelovima edukacijske makete. Definiran je način komunikacije između podređenog i nadređenog kontrolera te opisan odabir, razvoj i upute za korištenje grafičkog korisničkog sučelja.



Slika 1: Ilustracija korištenja edukacijske makete

2 Upravljački uređaji

Upravljački uređaj je stroj koji na temelju unutarnje logike i ulaznih signala upravlja procesom putem odgovarajućih izvršnih signala, odnosno izvršnih elemenata (aktuatora). Unutarnja logika stroja je skup radnji koje stroj izvršava pod točno određenim uvjetima, uvjetovanim ulaznim signalima, i na točno određeni način (na primjer Wattov centrifugalni regulator protoka pare prikazan na slici 2). Signali su energetski nosioci informacije koji mogu biti mehanički, električni, elektromagnetski, kemijski ili biološki. Vrsta izlaznog signala, odnosno rezultat unutarnje logike, može se razlikovati od vrste ulaznog signala.



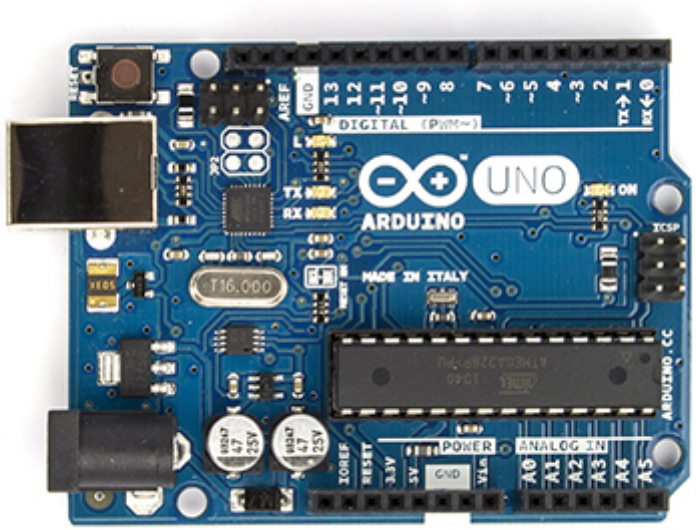
Slika 2: Centrifugalni regulator protoka pare - unutarnja logika je mehanička izvedba regulatora dok su ulazni (brzina vrtnje vratila) i izlazni signali (kut zakreta ventila) mehanički [4]

Upravljački uređaji korišteni u ovom radu su Arduino mikrokotroler i Raspberry Pi računalo. Pojednostavljeno, pod unutarnjom logikom mikrokontrolera može se smatrati program za mikrokontroler. Ulazni i izlazni signali su električne vrste. Pod ulaznim signalima smatraju se podatci potrebni za izvršavanje programa i očitanja stanja procesa pomoću senzora.
Arduino mikrokontroler je projekt otvorenog tipa (engl. *open source*) započet 2005. godine u Italiji s ciljem izrade mikrokontrolera koji bi cijenom bio pristupačan studentima. Osim cijenom, Arduino mikrokontroleri odlikuju se i jednostavnošću korištenja. Riječ je o platformi koja se nalazi na jednoj tiskanoj pločici i sadrži priključke za spajanje mikrokontrolera s ulazno/izlaznim jedinicama. Projekt je ubrzo nadišao granice Italije i u razno raznim inačicama našao svoje mjesto u brojnim hobističkim i edukacijskim projektima. Raspberry Pi nije mikrokontroler već mikroračunalo integrirano na jednoj tiskanoj pločici. Raspberry Pi zaklada osmislila je

računalo pristupačne cijene s ciljem promicanja računarstva dostupnog svima. Za upravljanje dijelovima edukacijske makete koristit će se Arduino Uno mikrokontroler kao podređeni mikrokontroler i Raspberry Pi Linux računalo kao nadređena upravljačka jedinica. Korisnik odabire željene parametre određenog procesa te preko serijske veze te podatke šalje Arduinu. Arduino Uno na temelju podataka dobivenih od korisnika i podataka prikupljenih preko senzora upravlja objektima regulacije.

2.1 Arduino Uno

Kao što je već spomenuto, za podređeni mikrokontroler koristit će se Arduino Uno, točnije Arduino Uno3[5]. Platforma je zasnovana na Atmel ATmega328 [6] integriranom sklopu. Tehničke specifikacije Arduino Uno3 mikrokontrolera prikazane su u tablici 1.



Slika 3: Arduino Uno3 [5]

Za komunikaciju s nadređenim kontrolerom koristit će se USB priključak, koji osim za komunikaciju može služiti i kao izvor napajanja. Raspberry Pi mikroračunalo razmjerno je osjetljivo na varijacije napona napajanja, pa kao izvor napajanja Arduino mikrokontrolera koristiti će se zasebni AC/DC adapter od $9 - 12\,V$. Digitalni priključci 0 i 1 služe za serijsku komunikaciju i njihova stanja se mijenjaju prilikom serijske komunikacije preko USB-a. Zbog toga se navedeni priključci neće koristiti za spajanje s elementima makete.

Tablica 1: Tehničke specifikacije Arduino Uno3 mikrokontrolera [5]

Mikrokontroler	ATmega328
Radni napon	$5\,V$
Preporučeni ulazni napon	$7 - 12\,V$
Granične vrijednosti ulaznog napona	$6 - 20\,V$
Polaritet vanjskog adaptera	središnji dio pozitivan
Broj digitalnih izlaza	14 (od toga 6 PWM)
Broj analognih ulaza	6
Jakost struje po ulazu/izlazu (engl. *source/sink*)	$40\,mA$
Flash memorija	$32\,kB$
SRAM	$2\,kB$
EEPROM	$1\,kB$
Radni takt	$16\,MHz$
Dimenzije	$68,6\,x\,53,4\,mm$
Masa	$25\,g$

2.1.1 Digitalni ulazi/izlazi

Digitalni priključci definiraju se kao ulazni ili izlazni unutar programa za mikrokontroler. Vrijednost očitavanja digitalnog ulaza ovisi o nazivnom naponu (V_{cc}) kojim se napaja ATmega328 mikroprocesorski integrirani krug u sklopu Arduino mikrokontrolerske platforme. Tablica 2 prikazuje raspon napona na ulazu i odgovarajuće vrijednosti koje očitava mikrokontroler [6].

Tablica 2: Raspon vrijednosti napona za digitalno očitavanje stanja [6]

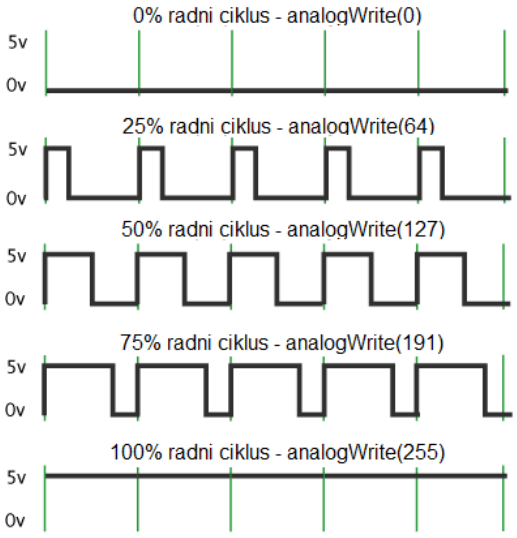
Očitana vrijednost	Raspon vrijednosti napona na ulazu
0	$-0,5 - 0,3\,V_{cc}$
1	$0,6\,V_{cc} - V_{cc} + 0,5$

Za granične vrijednosti napona, kao i u slučaju da priključak nije spojen preko otpornika (engl. *pull-down*) na referentnu točku od $0\,V$, vrijednosti očitanja mogu stohastički varirati između 0 i 1. Ukoliko je digitalni priključak postavljen kao digitalni izlaz, tada je njegovo visoko stanje odgovara naponskom stanju $+5\,V$ (spajanje na pozitivni polaritet napajanja), dok nisko stanje odgovara spajanju na referentnu točku od $0\,V$.

2.1.2 Impulsno-širinska modulacija

Impulsno-širinska modulacija (engl. ***P**ulse **W**idth **M**odulation*, **PWM**) je postupak emuliranja analognih veličina primjenom digitalnih (dvo-razinskih) valnih oblika. Priključak koji podržava impulsno-širinsku modulaciju drži se u visokom stanju određeni interval vremena, te se potom priključak vraća u nisko stanje. Vremenski interval između rastućeg i padajućeg brida impulsa naziva se širina impulsa (T). Vremenski interval između dva rastuća ili dva padajuća brida impulsa naziva se period (P). Širina impulsa proporcionalna je amplitudi željenog analognog signala. Arduino Uno3 raspolaže sa šest impulsno-širinskih digitalnih izlaza koji podržavaju PWM modulaciju digitalnog signala, a koji rade na frekvenciji od $500\,Hz$. Radni ciklus (engl. *duty cycle*) je postotak vremena visokog stanja u jednoj periodi i opisan je sljedećom jednadžbom:

$$D = \frac{T}{P} \cdot 100 \quad [\%] \tag{1}$$



Slika 4: Radni ciklus impulsno-širinske modulacije [2]

Pulsno širinski izlazi mogu se koristiti za generiranje referentog signala proizvoljnog oblika ili za okidanje tranzistora u ulozi sklopke.

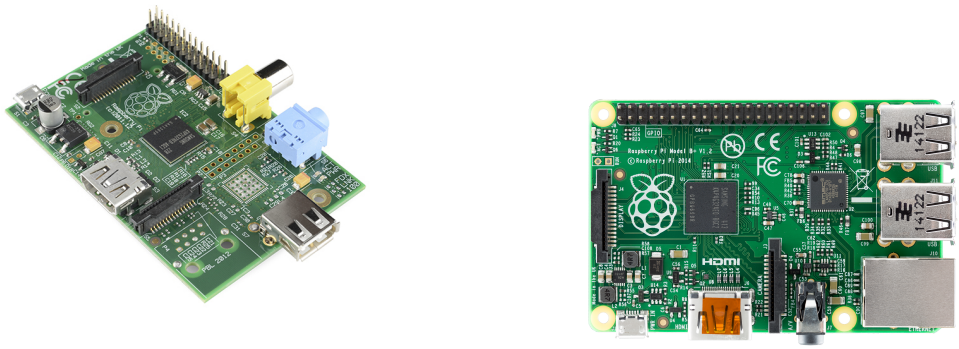
2.1.3 Analogni ulazi

Analogni ulaz je 10-bitni analogno-digitalni pretvornik. To znači da će očitana analogna vrijednost od $0-5\,V$ biti podijeljena na 1024 binarne kombinacije, gdje najniži bit binarnog podatka

odgovara vrijednosti od približno $4,88\,mV$ pri napajanju od $5\,V$. Naime, rezolucija analogno-digitalnog konvertera ovisi o referentnom naponu s kojom Arduino uspoređuje ulazni signal. Za referentni napon mogu se koristiti ugrađene reference od $5\,V$ ili $1.1\,V$, ili preko **AREF** priključka spojiti proizvoljan eksterni referentni napon, ali do iznosa $5\,V$. Prilikom korištenja eksternog referentnog napona različitog od $5\,V$ potrebno je uzeti u obzir da prvo očitanje možda neće biti ispravno. Do pogrešnog očitanja dolazi zbog fluktuacije napona između ulaznog priključka i vanjske reference napona uzrokovane razlikom napona između dva priključka[6].

2.2 Raspberry Pi

Za potrebe ovog diplomskog rada korišteno je Raspberry Pi Model B[7] računalo. Model B je unaprijeđeno izdanje prvobitnog Raspberry Pi mikroračunala, sada poznatijeg pod imenom Raspberry Pi Model A. Model B ima Ethernet priključak i dvostruko veću radnu memoriju (512 naspram $256\,MB$), kao i dva USB priključka naspram samo jednog na Modelu A (slika 5). U trenutku pisanja ovog teksta postoje još i Raspberry Pi Model B+, i najnoviji dodatak Raspberry Pi obitelji, Raspberry Pi 2 Model B.Tehničke specifikacije Raspberry Pi Model B (dalje u tekstu samo Raspberry Pi) dane su u tablici 3.



(a) Raspberry Pi Model A (b) Raspberry Pi Model B

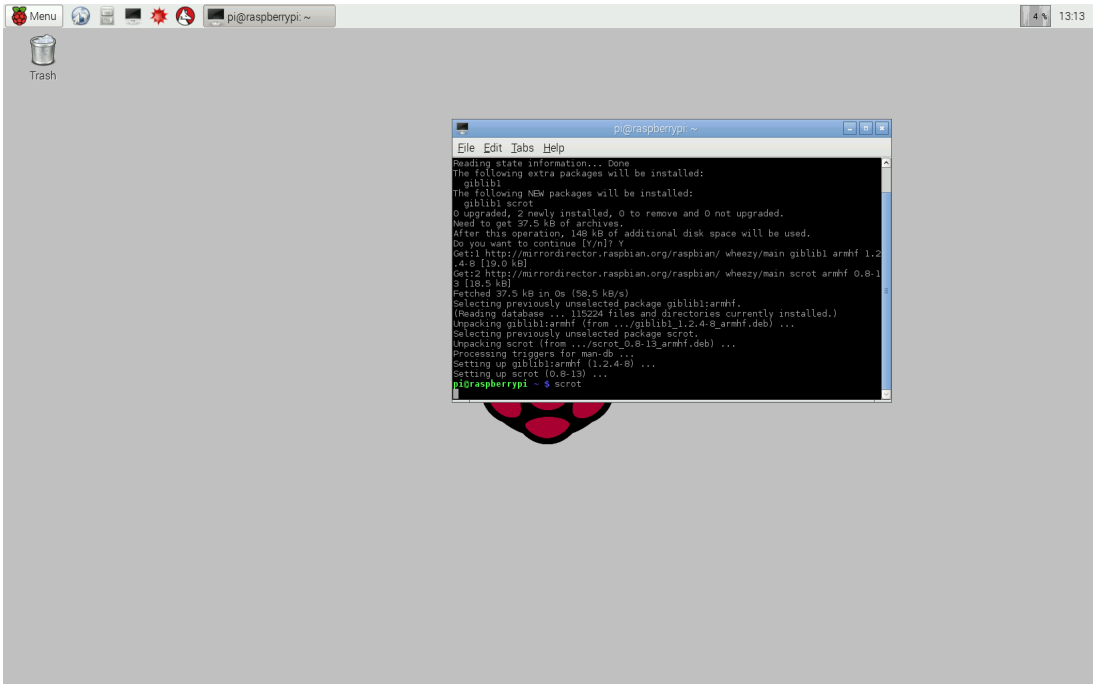
Slika 5: Usporedba Raspberry Pi Modela A i Modela B

Raspberry Pi ima mogućnost pokretanja operacijskog sustava baziranog na Linux jezgri (engl. *kernel*). Od ponuđenih distribucija Linuxa odabran je operacijski sustav Raspbian[9]. Raspbian je baziran na Debian sustavu i optimiziran za rad s Raspberry Pi arhitekturom. Prednost Raspbiana nad drugim Linux distribucijama je to što je korisnik odmah spreman na rad jer brojni programi dolaze instalirani. Raspberry Pi raspolaže s dva USB priključka što je dovoljno da se na njega priključe osnovne ulazne periferije kao što su miš i tipkovnica. Za potrebe serijske ko-

munikacije s podređenim kontrolerom potrebno je koristiti USB razdjelnik, kako bi istovremeno korisnik mogao koristiti periferije i ostvariti komunikaciju. Zbog izrazite osjetljivosti Raspberry Pi-ja na varijacije napona uslijed opterećenja, poželjno je koristiti USB razdjelnik sa zasebnim izvorom napajanja. Slika 6 prikazuje dva moguća načina rada u Raspbian operacijskom sustavu, preko konzole i preko grafičkog *desktop* sučelja.

Tablica 3: Tehničke specifikacije Raspberry Pi Model B računala [8]

	Raspberry Pi Model B
SoC (engl. system on a chip):	Broadcom BCM2835
CPU (procesor):	$700\,MHz$ ARM1176JZF-S
GPU (grafički procesor):	Broadcom
Memorija (SDRAM) :	$512\,MB$
USB utičnice :	2
Video ulaz :	15-pin MIPI Camera Interface
Video izlaz :	HDMI $1920x1200$
Audio izlaz :	$3.5\,mm$ audio konektor
Ethernet priključak :	$10/100\,Mbit/s$ Ethernet
Lokalna pohrana	SD/MMC kartica
Ulazno/izlazni priključci :	17 ulazno/izlaznih priključaka
Izvor napajanja :	$5\,V$ preko microUSB-a
Dimenzije :	$85,60x56,5$
Masa :	$45\,g$

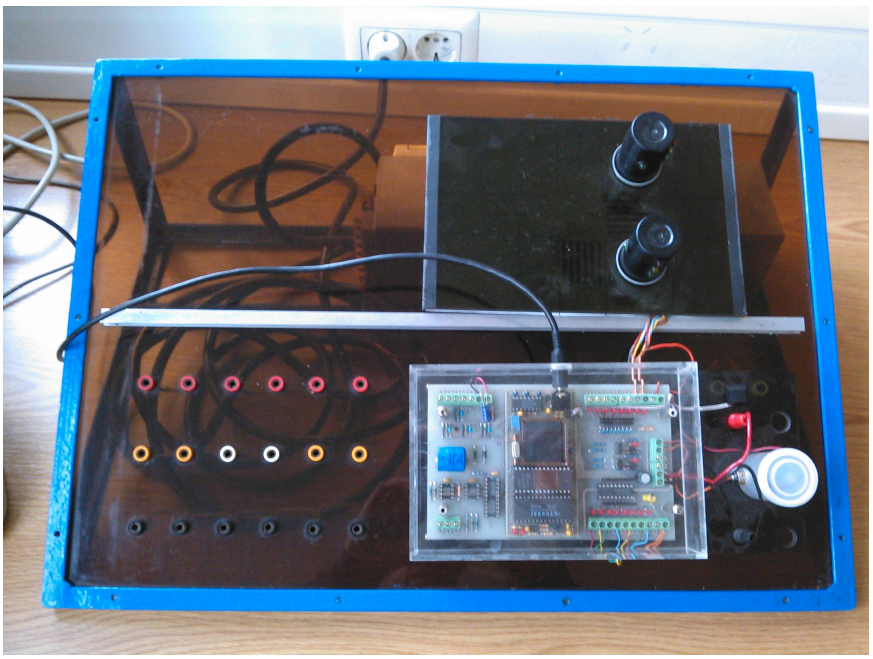


Slika 6: Raspberry Pi X grafičko sučelje s terminalom

3 Analiza objekata upravljanja

Analiza objekta regulacije sastoji se od određivanja načina rada pojedinog objekta i potrebne interakcije s podređenim mikrokontrolerom. Na temelju podataka dobivenih analizom svakog pojedinog objekta može se pristupiti izradi njegovog simulacijskog modela za implementaciju u grafičkom sučelje nadređenog kontrolera. Princip rada pojedinog objekta je bitan za izradu algoritma programa mikrokontrolera koji će direktno preko svojih ulaza i izlaza upravljati objektima edukacijske makete. Edukacijska maketa, prikazana na slici 7, sastoji se od tri međusobno neovisna podsustava kojima je potrebno upravljati:

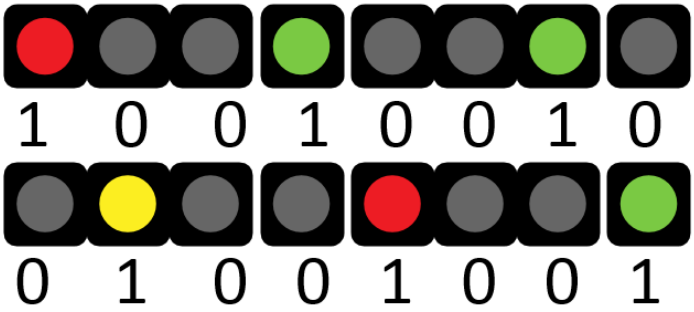
- podsustav raskršća sa semaforima, gdje je potrebno realizirati sekvencijalno upravljanje signalnim svjetlima semafora
- emulator rada programabilnog logičkog kontrolera (PLC-a) s osnovnim logičkim funkcionalnostima
- toplinska komora s grijačem i senzorom temperature kao pokazna maketa sustava upravljanja (regulacije) toplinskih procesa



Slika 7: Edukacijska maketa

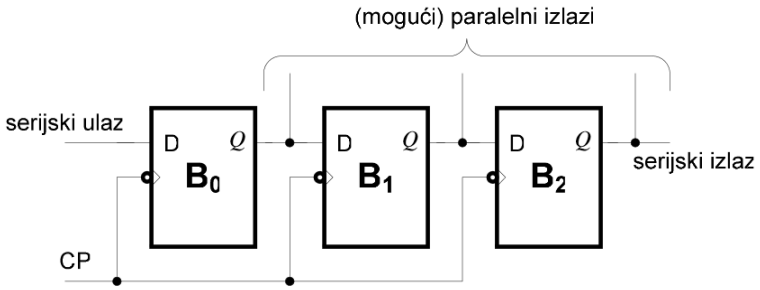
3.1 Sekvencijalno upravljanje semaforima

Sekvencijalno upravljanje semafora dio je funkcionalnosti izvorne edukacijske makete koji simulira upravljanje prometnim svjetlima na raskrižju. Maketa se sastoji od dva semafora za vozila i dva semafora za pješake. Semafori za vozila sastoje se od tri indikacijske svjetleće diode (crvena, žuta i zelena), dok semafori za pješake posjeduju samo jednu indikacijsku svijetleću diodu.Semafor se može nalaziti samo u jednom stanju, istovremeno može svjetliti samo jedna od tri indikacijske svijetleće diode na semaforu za vozila, odnosno zeleno svjetlo na semaforu za pješake može biti uključeno ili isključeno. Stanje svake diode semafora možemo zapisati korištenjem digitalne logike, gdje vrijednost 1 odgovara stanju uključenja diode dok vrijednost 0 odgovara stanju isključene svijetleće diode. Za izmjenu stanja svakog signalnog



Slika 8: Logičke vrijednosti stanja semafora

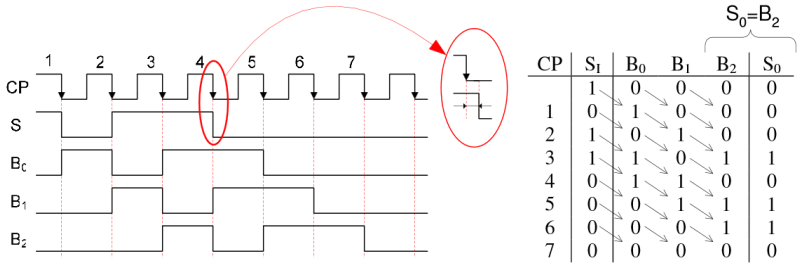
svjetla potrebno je osam digitalnih izlaza, jedan za svaku svjetlosnu diodu. Kako je ukupni broj digitalnih izlaza na Arduino mikrokontroleru razmjerno mali, umjesto izravnog spajanja na osam digitalnih izlaza mikrokontrolera koristit će se serijsko-paralelni (engl. **SIPO**, ***S**erial **I**n - **P**arallel **O**ut*) posmačni registar (engl. *shift register*). Posmačni registar radi na principu pomicanja bit podatka od ulaza prema izlazu. Sastoji se od niza serijski spojenih bistabila gdje je izlaz iz jednog bistabila ulaz u drugi.



Slika 9: Posmačni registar - serijski spojeni bistabili [10]

Pomicanjem svakog registra automatski se mijenja stanje na paralelnom izlazu, kako je ilus-

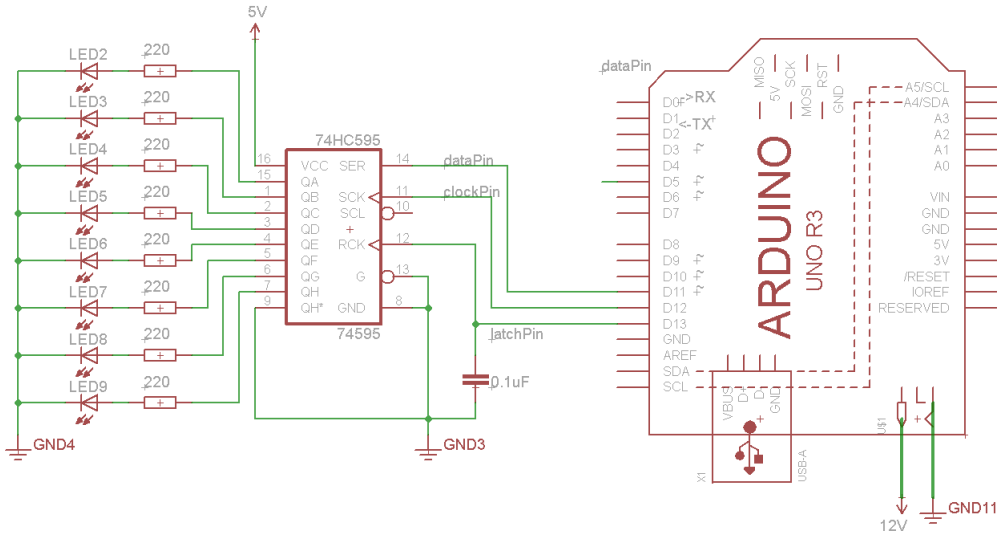
trirano na slici 10. Svi serijski spojeni bistabili dijele isti sistemski sat, kojeg određuje mikrokontroler ili zasebni kvarcni kristal. Ovisno o izvedbi registra, podatak se pomiče na uzlaznom ili silaznom bridu napona taktnog signala posmačnog registra (engl. *storage register clock*). Slika 11 prikazuje način spajanja makete sekvencijalnog semafora s Arduino mikrokontrole-



CP	S_I	B_0	B_1	B_2	S_0
	1	0	0	0	0
1	0	1	0	0	0
2	1	0	1	0	0
3	1	1	0	1	1
4	0	1	1	0	0
5	0	0	1	1	1
6	0	0	0	1	1
7	0	0	0	0	0

Slika 10: Posmačni registar - pomicanje podataka kroz registar [10]

rom. Korišteni posmačni registar je integrirani krug 74HC595 [11] 8-bitni registar upravljan s tri ulazna digitalna priključka. Pritom se takozvani *latch* ulaz primjenjuje za sprječavanje promjene stanja na paralelnim izlazima prilikom pomicanja bitova unutar registra. Podatci se prvo zapisuju u ulazni spremnik, potom se promjenom stanja *latch* ulazu kopiraju u izlazni spremnik i dolazi do trenutne promjene stanja paralelnih izlaza. Upotrebom posmačnog registra umjesto osam digitalnih izlaza iskorištena su samo tri.



Slika 11: Shema spajanja makete semafora na mikrokontroler

3.2 Programabilni logički kontroler

Programabilni logički kontroler (dalje u tekstu samo PLC, engl. **PLC** - ***P****rogrammable* ***L****ogic* ***C****ontroller*) je integrirano industrijsko računalo koje sadrži sve potrebne elemente za praćenje stanja i upravljanje raznim procesima, prvenstveno onima koji su karakterizirani logičkim stanjima. PLC na edukacijskoj maketi čini Arduino mikrokontroler s programom koji simulira rad temeljne izvedbe PLC-a. Program simulacije rada PLC izvodi se ciklički i sastoji se od tri faze:

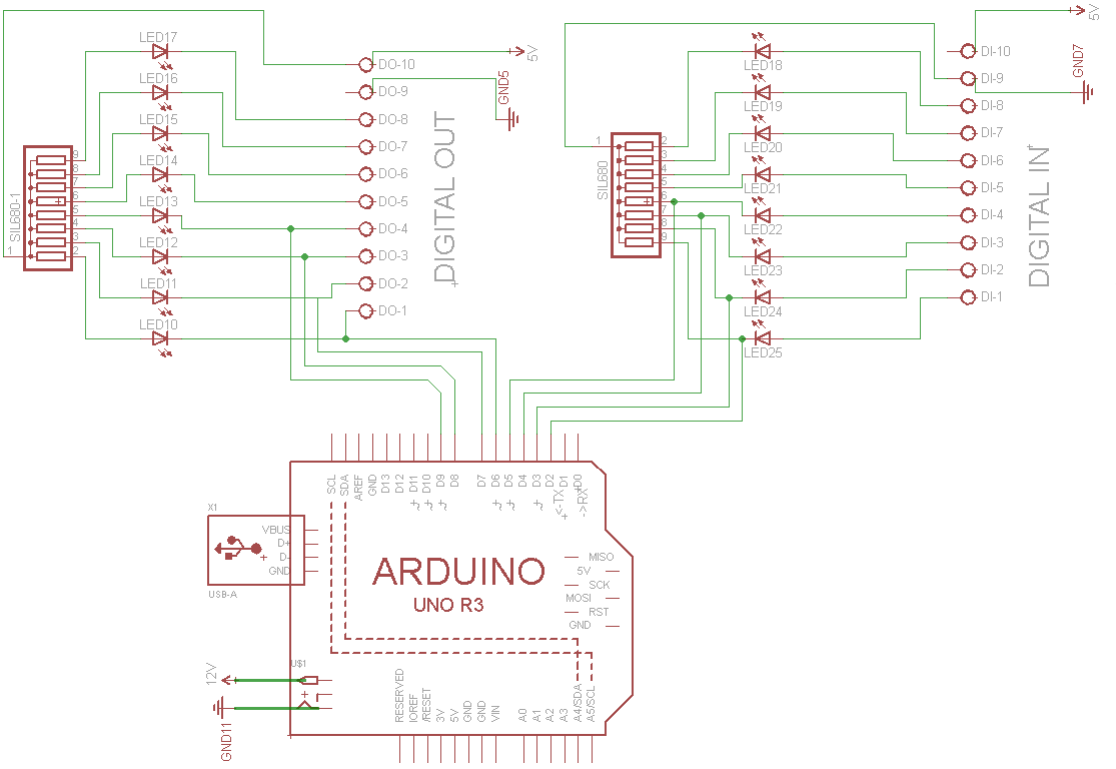
- čitanje ulaznih logičkih stanja
- izvršavanje programskog koda
- ispisivanje rezultata logičkih operacija na logičke (digitalne) izlaze

Stanje logičkog ulaza mijenja se spajanjem ili odspajanjem naponske razine od $+5\,V$ na ulaznoj liniji. Ako je ulaz spojen na izvor napona, vrijednost na digitalnom ulazu iznosit će logički '1', odnosno ako je spojen na nisko naponsko stanje, logičko stanje je tada '0'. Stanja izlaznih varijabli mijenjaju se ovisno o zadanim logičkim operacijama. Na svakom ulazu i izlazu iz mikrokontrolera nalaze se svjetlosne diode koje služe kao indikacijska svjetla o trenutnom stanju određenog ulaza ili izlaza. Svekupno je na izvornoj maketi bilo predviđeno šestnaest digitalnih linija, osam ulaznih i osam izlaznih. To je više nego što ih Arduino Uno3 ima i zbog toga broj ulaza i izlaza ograničava se na osam, od kojih četiri čine ulazi a četiri izlazi iz mikrokontrolera. Shema spajanja ulaza i izlaza makete s Arduino Uno3 mikrokontrolerom prikazana je na slici 12.

Temelj logičkih operacija predstavlja Booleova algebra. Osnovni element logičke algebre je logička izjava. Za svaku izjavu može se jednoznačno utvrditi da je istinita ili lažna. Ako je izjava istinita ona ima vrijednost '1', a ako je neistinita ima vrijednost '0'. Osnovne logičke operacije su **I**, **ILI** i **NE**. Sve ostale logičke operacije mogu se svesti na kombinaciju osnovnih operacija.

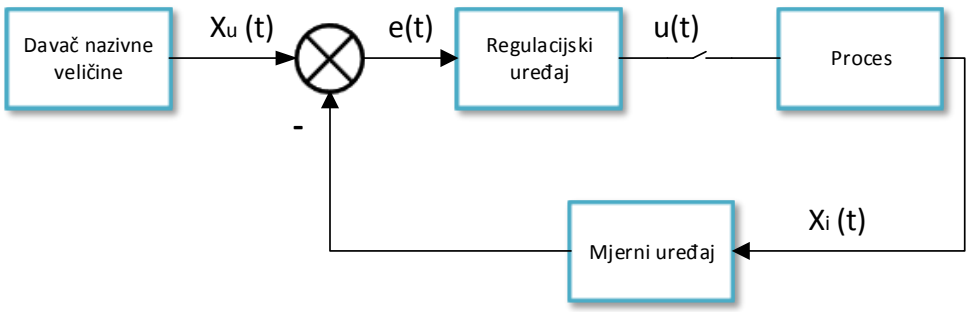
3.3 Automatska regulacija toplinskog procesa

Automatska regulacija je postupak vođenja procesa po načelu negativne povratne veze bez prisustva čovjeka. Djelovanje regulatora zasniva se na razlici između ulazne referentne veličine i upravljačke veličine. Temeljem regulacijskog odstupanja određuje se upravljačka veličina kojom izvršni član djeluje na proces. Ovisno o referentnoj veličini bira se i vrsta regulacije.



Slika 12: Shema spajanja ulaza i izlaza PLC-a na mikrokontroler

Čvrsta regulacija je regulacija s vremenski sporo promjenjivom ili vremenski-invarijantnom (čvrstom) referentnom veličinom. Koristi se kako bi se uklonilo djelovanje poremećaja na reguliranu veličinu. Slijedna regulacija je je regulacija s vremenski izrazito varijantnom referentnom veličinom. Cilj regulacije procesa je da proces dostigne željenu referentnu veličinu, odnosno da regulacijsko odstupanje bude jednako nuli. Regulacijski krug se sastoji od davača nazivne veličine, komparatora, regulacijskog uređaja (regulatora), objekta regulacije (proces) i mjernog uređaja.



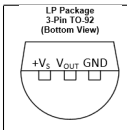
Slika 13: Zatvoreni regulacijski krug

Slika 13 prikazuje pojednostavljeni prikaz regulacijskog kruga. Veličine sa slike su :

- $x_u(t)$ - referentna veličina
- $u(t)$ - upravljačka veličina
- $x_i(t)$ - regulirana veličina
- $e(t)$ - regulacijsko odstupanje, razlika između referentne i regulirane veličine

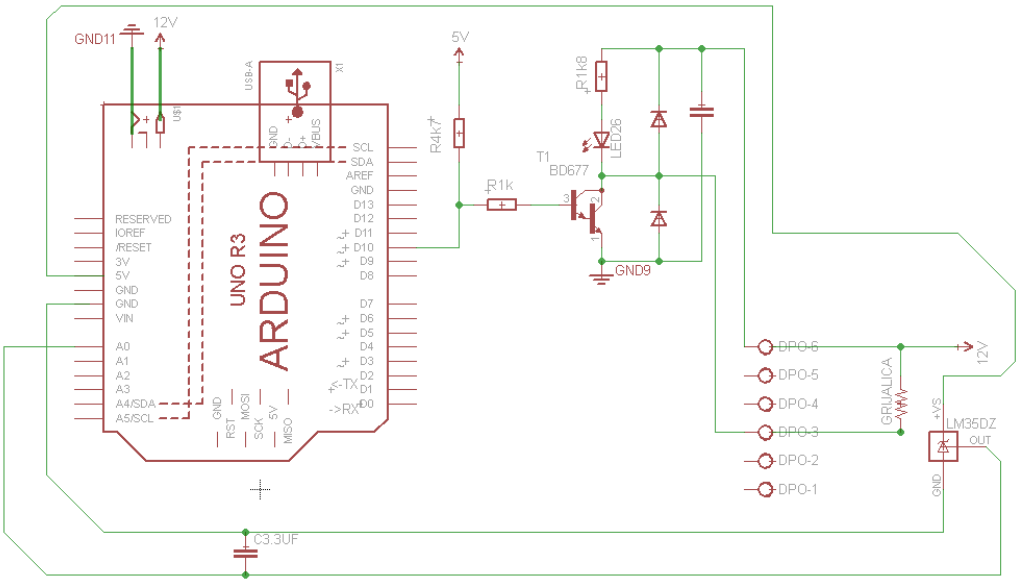
Automatska regulacija toplinskog procesa odnosi se na regulaciju temperature unutar toplinske komore na edukacijskoj maketi (slika 7). Toplinska komora sastoji se od plastičnog cilindra unutar kojeg se nalazi grijaće tijelo i temperaturni senzor. Kao grijaće tijelo korištena je žaruljica nazivne snage od $2.4\,W$ spojena na zasebni strujni krug napona napajanja $12\,V$. Zasebni strujni krug se uključuje ili isključuje pomoću bipolarnog tranzistora čija je baza spojena na digitalni PWM izlaz mikrokontrolera. Temperaturni senzor je LM35DZ [13] u TO-92 izvedbi. Senzor ne zahtijeva kalibraciju niti dodatno elektorničko sklopovlje, već se može spojiti direktno na Arduinov analogni ulaz. Najvažnije specifikacije LM35DZ senzora dane su u tablici 4. Senzor

Tablica 4: Specifikacije temperaturnog senzora

	Iznos
Osjetljivost	$10\,mv/^oC$
Preciznost	$0.5\,^oC$
Mjerni raspon	od -55 do $+150^oC$
Radni napon	od 4 do $30\,V$

je relativno neosjetljiv na šum, no kako bi se eliminirao visokofrekvencijski šum uzrokovan elektromagnetskim smetnjama na priključcima, između izlaznog priključka i referentne točke (uzemljenja) stavlja se $3.3\mu F$ kondenzator. Za referentnu veličinu napona analognog ulaza koristit će se interni napon od $1.1V$. Odabir referentnog napona od $1.1V$ ograničava maksimalnu temperaturu koja se može očitati na 110^oC, što je znatno viša temperatura od one koju termokomora može postići. Shema spajanja edukacijske makete na Arduino mikrokontroler prikazana je na slici 14.

Referentnu veličinu i parametre sustava određuje korisnik i preko nadređenog kontrolera zadaje regulatoru. Referentna veličina je vremenski invarijantna, odnosno primjenuje se koncept čvrste regulacije procesa. Regulirana veličina je temperatura zraka unutar termokomore. Regulator reguliranu veličinu oduzima od referentne kako bi odredio regulacijsko odstupanje. Prema

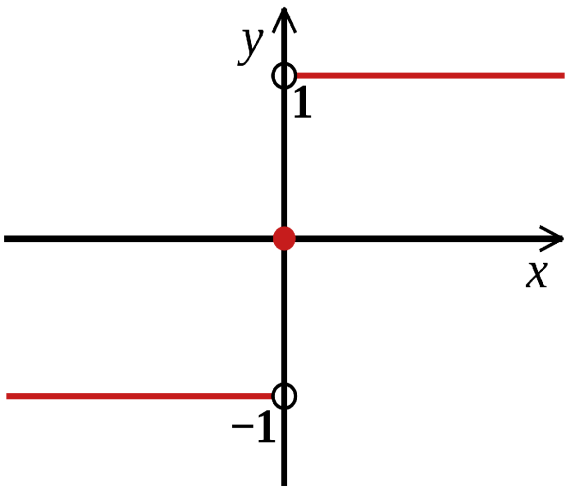


Slika 14: Shema spajanja toplinske komore s mikrokontrolerom

određenom algoritmu, regulator određuje izlaznu, upravljačku vrijednost koja okida tranzistor i uključuje ili isključuje grijaće tijelo i na taj način djeluje na reguliranu veličinu. Arduino mikrokontroler sadrži dva oblika regulacije toplinskog procesa, relejnu dvopoložajnu regulaciju (engl. *On/Off* ili *bang-bang*) i PID regulaciju.

3.3.1 Relejna dvopoložajna regulacija

Relejna dvopoložajna regulacija primjenjuje takozvani dvopoložajni regulator (slika 15) u kojem se regulirani proces upravlja uključivanjem ili isključivanjem upravljačke veličine. Dvopoložajni

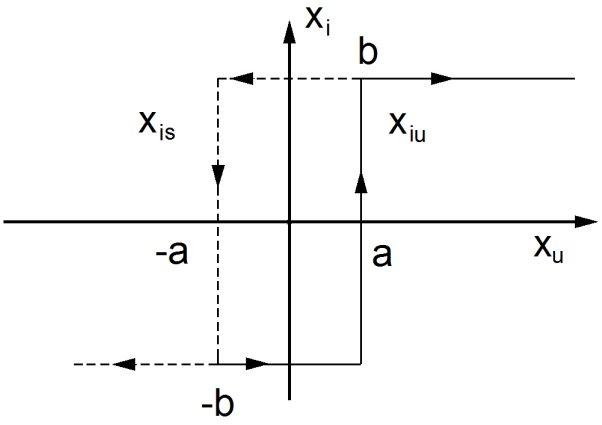


$$f(x) = \begin{cases} 1 & \text{za} \quad x > 0 \\ 0 & \text{za} \quad x = 0 \\ -1 & \text{za} \quad x < 0 \end{cases}$$

Slika 15: Dvopoložajni regulator - signum funkcija

relejni regulatori često su implementirani s takozvanom histerezom, koja omogućuje da se sta-

nje izlaza regulatora ne mijenja samo sa predznakom signala regulacijskog odstupanja, već se dopušta određeni hod oko ravnotežnog stanja (regulacijskog odstupanja jednakog nuli), kako dvopoložajni regulator ne bi prečesto mijenjao stanje. Karakteristika dvopoložajnog releja s histerezom prikazana je na slici 16. Područje između $-a$ i a je raspon histereze. Pri povećanju



Slika 16: Dvopoložajni regulator s histerezom [14]

vrijednosti ulazne funkcije x_u iznad nultočke dolazi do okidanja releja tek kod $x_u = a$. Pri smanjenju vrijednosti x_u ispod nultočke dolazi kod $x_u = -a$ do obratnog okidanja. Za uzlaznu granu vrijedi sljedeće :

$$x_{iu} = \begin{cases} -b & \text{za} \quad x_u < a \\ b & \text{za} \quad x_u > a \end{cases}$$

Za silaznu granu vrijedi sljedeće :

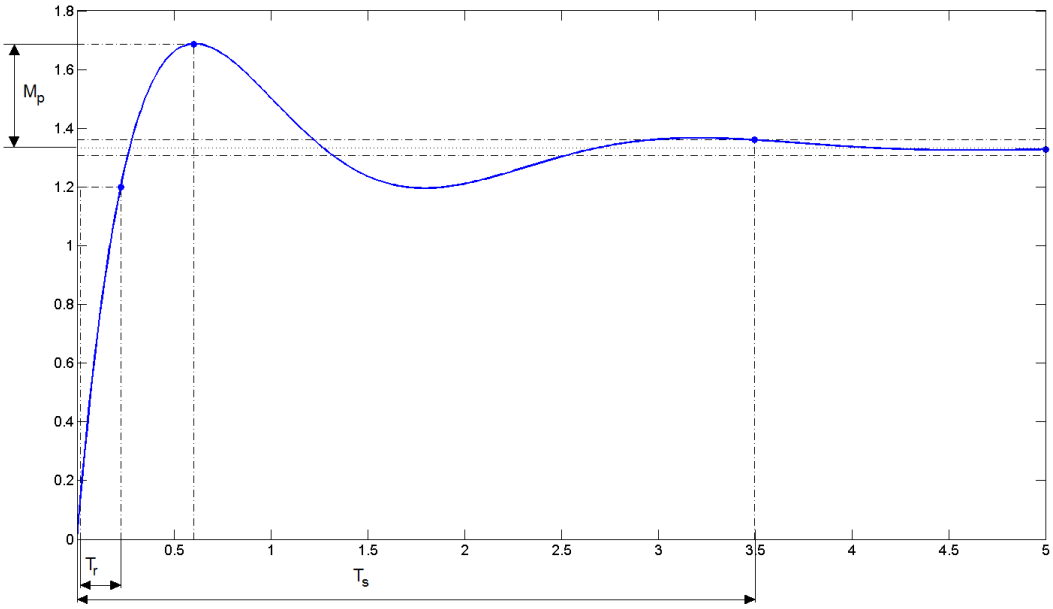
$$x_{is} = \begin{cases} -b & \text{za} \quad x_u < -a \\ b & \text{za} \quad x_u > -a \end{cases}$$

Zasebni strujni krug toplinskog procesa bit će zatvoren sve dok je mjerna veličina manja od referentne veličine. Ako pak mjerna veličina prevaziđe referentnu za polovicu širine histereze, slijedi isključivanje i postupno hlađenje toplinske komore sve dok regulirana veličina ne padne ispod vrijednosti referentne veličine umanjene za polovicu širine histereze, nakon čega se grijač ponovno uključuje. Nakon što se isključi strujni krug grijaćeg tijela, žaruljica prestaje svjetliti, no ona je još uvijek spremnik topline čija je temperatura veća od temperature zraka unutar komore. Temperatura zraka unutar komore raste sve dok se toplinska energija žaruljice ne disipira u okoliš.

3.3.2 PID regulator

PID regulator je najzastupljeniji oblik regulatora za realne procese i zasniva se na proporcionalnom, integralnom i derivacijskom djelovanju na regulacijsko odstupanje od referentne veličine. Zahtjevi na kvalitetu odziva regulacijskog kruga s PID regulatorom u vremensko kontinuiranom području su:

- T_r - vrijeme porasta, vrijeme potrebno da odziv sustava poraste od 10% do 90% konačne vrijednosti
- T_s - vrijeme smirivanja, vrijeme potrebno da odziv sustava doesgne i zadrži unutar zadanog intervala konačne vrijednosti
- M_p - postotni prebačaj, maksimalna amplituda prigušenih oscilacija
- e_0 - trajno regulacijsko odstupanje od referentne veličine



Slika 17: Zahtjevi na kvalitetu odziva

Djelovanje proporcionalnog člana u vremenski kontinuiranom području definirano je sljedećom jednadžbom:

$$u_P(t) = K_p\, e(t) \tag{2}$$

gdje je K_p proporcionalno pojačanje regulatora. Proporcionalni član djeluje proporcionalno iznosu regulacijskog odstupanja te zato ne može eliminirati trajno regulacijsko odstupanje. Porastom pojačanja K_p vrijeme porasta se smanjuje, ali se prebačaj povećava.

Djelovanje integralnog člana u vremenski kontinuiranom području definirano je sljedećom jednadžbom:

$$u_I(t) = K_i \int_0^t e(t)dt \tag{3}$$

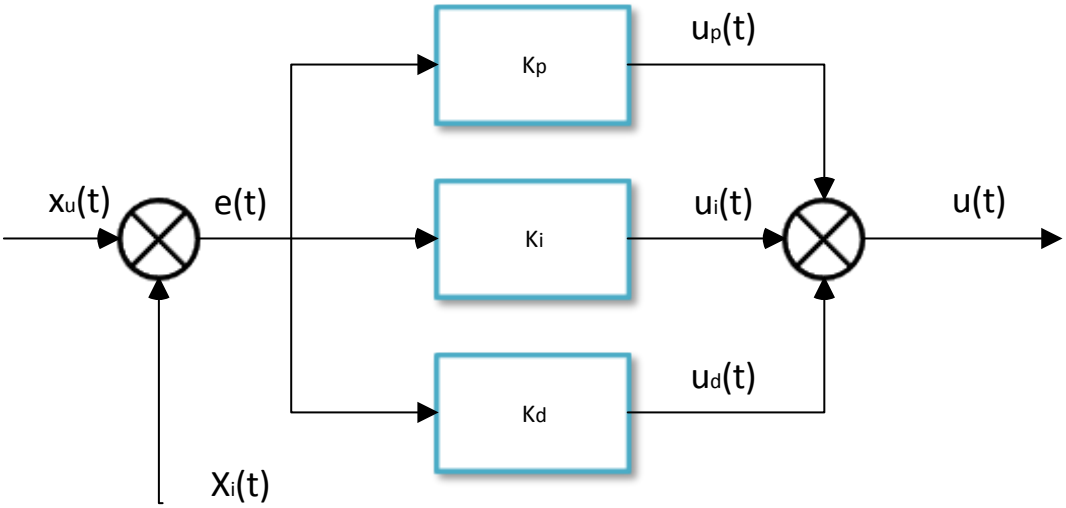
gdje je K_i intgeralno pojačanje regulatora. Integracijski član djeluje proporcionalno vremenskom integralu regulacijskog odstupanja (akumulira prethodna stanja) čime unosi kašnjenje u sustav ali zato može ukloniti stacionarno regulacijsko odstupanje. Porastom pojačanja K_i smanjuje se vrijeme porasta ali se povećava prebačaj i vrijeme smirivanja.

Derivacijski član u vremenskom području definiran je sljedećom jednadžbom:

$$u_D(t) = K_d \frac{de(t)}{dt} \tag{4}$$

gdje je K_d derivacijsko pojačanje regulatora. Derivacijski član djeluje unaprijedno (prediktivno) na iznos regulacijskog odstupanja. Povećanjem pojačanja K_d smanjuje se prebačaj i vrijeme smirivanja.

Slika 18 prikazuje opću paralelnu formu regulatora koja će se koristiti za regulaciju toplinskog procesa.



Slika 18: Opća paralelna forma regulatora

Djelovanje PID regulatora na upravljačku veličinu s obzirom na parametre sustava i regulacijsko

odstupanje u vremenski kontinuiranom području prikazano je jednadžbom 5.

$$u(t) = K_p\, e(t) + \, K_i \int_0^t e(t)dt + \, K_d \, \frac{de(t)}{dt} \tag{5}$$

Mikrokontroler kao regulator procesa ne može obrađivati analogne signale sa senzora u vremenski kontinuiranom području. Podatke dobivene sa senzora potrebno je pretvoriti u digtalnu vrijednost pomoću analogno-digitalnog pretvornika. Stoga, za implementaciju PID regulatora opisanog jednadžbom 5 potrebno je izraz prebaciti iz vremenski kontinuiranog područja u vremenski diskretno područje. Diskretni signal dobiva se uzorkovanjem kontinuiranog signala s određenim vremenom uzorkovanja T_0. Odnos između kontinuiranog vremena i vremena uzorkovanja dan je jednadžbom 6.

$$t = k\, T_0 \quad t \in \mathbb{R}, k \in \mathbb{Z} \tag{6}$$

Izraz 2 u vremenski diskretnom području ima sljedeći oblik:

$$u_P(k) = K_p\, e(k) \tag{7}$$

Djelovanje vremenski kontinuiranog integralnog člana u vremenski diskretnom području aproksimiran je sumom površine od početka uzorkovanja do trenutnog člana u nizu.

$$\int_0^t e(t) \cong T_0 \sum_{k=0}^{k-1} e(k) \tag{8}$$

Izraz 3 u vremenski diskretnom području ima sljedeći oblik:

$$u_I(k) = u_I(k-1) + K_i\, T_0\, e(k) \tag{9}$$

Diskretizacija derivacijskog djelovanja u kontinuiranom području temelji se na numeričkoj aproksimaciji diferencijalne jednadžbe Eulerovom unazadnom diferencijom.

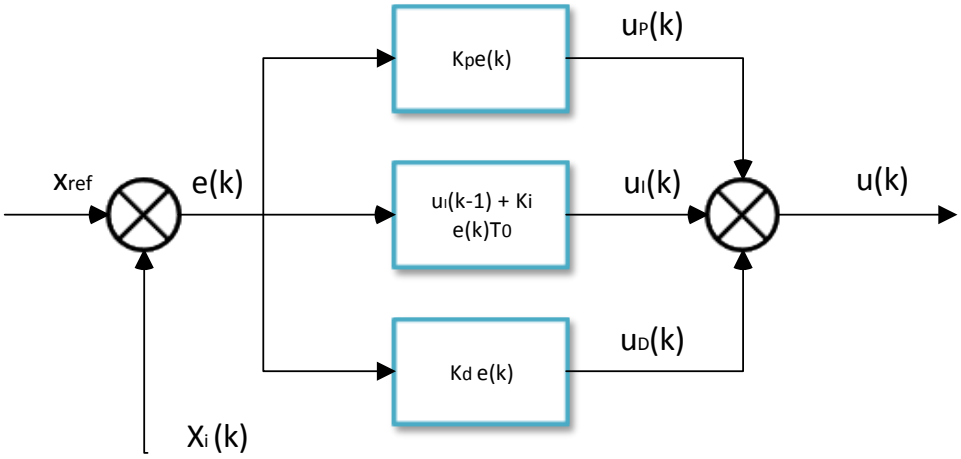
$$\frac{de(t)}{dt} \cong \Delta e(k) = \frac{e(k) \, - \, e(k-1)}{T_0} \tag{10}$$

Izraz 4 u vremenski diskretnom području ima sljedeći oblik:

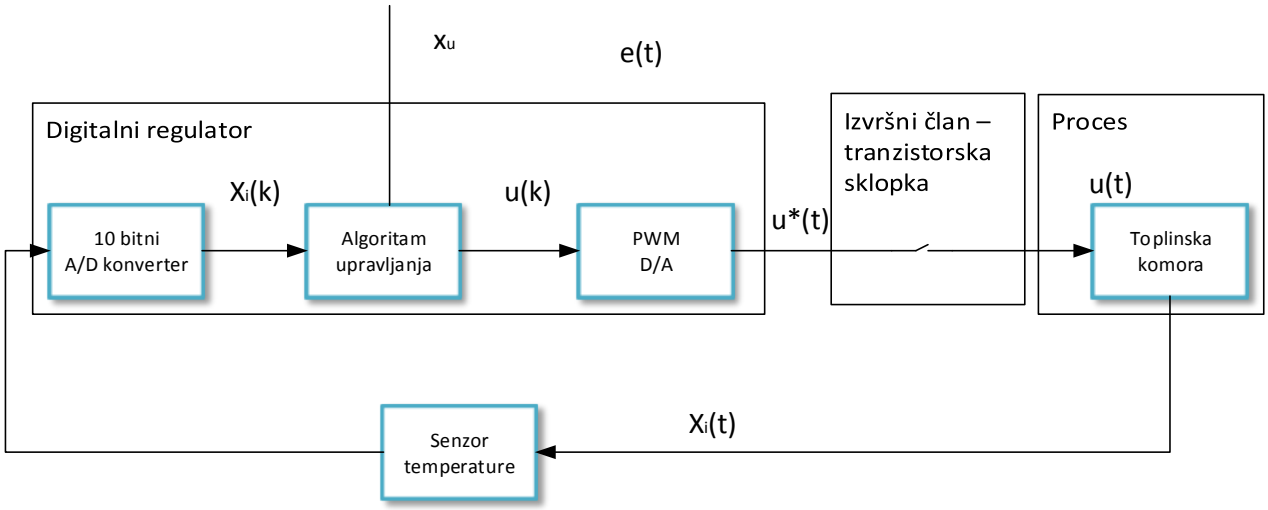
$$u_D(k) = K_d \frac{e(k) \, - \, e(k-1)}{T_0} \tag{11}$$

Algoritam upravljanja PID regulatora u vremenski diskretnom području [15] definiran je izrazom 12 i prikazan slikom 19.

$$u(k) = K_p\, e(k) + u_I(k-1) + K_i\, T_0\, e(k) + K_d \frac{e(k) - e(k-1)}{T_0} \tag{12}$$



Slika 19: Paralelni PID regulator u vremenski diskretnom području



Slika 20: Vremenko diskretni regulacijski krug

Slika 20 prikazuje zatvoreni regulacijski krug za regulaciju procesa u vremenski diskretnom području. Prije A/D pretvornika nema pretvornika jer se usporedba referentne vrijednosti i regulirane veličine vrši digitalno, unutar programa mikrokontrolera. Iznos upravljačke veličine određuje radni ciklus PWM priključka (slika 4). Radni ciklus PWM priključka mikrokontrolera definiran je unutar intervala od $0-255$. Zbog toga se uvodi sljedeće ograničenje na upravljačku varijablu :

$$0 \leq u(k) \leq 255 \tag{13}$$

Za ispravan rad PID regulatora nije dovoljno limitirati izlaz regulatora, već i ograničiti stanje integratora. Integralna komponenta upravljačke veličine (u_I) raste svakim korakom uzorkovanja i ubrzo postiže vrlo velike iznose, što dovodi do velikog prebačaja i dugog vremena smirivanja. Ograničavanje stanja integratora postiže se upotrebom *reset antiwindup* metode gdje se stanje integratora resetira na vrijednost koja odgovara razlici gornje ili donje granice upravljačke varijable i proporcionalno-derivirajućeg djelovanja (jednadžba 14).

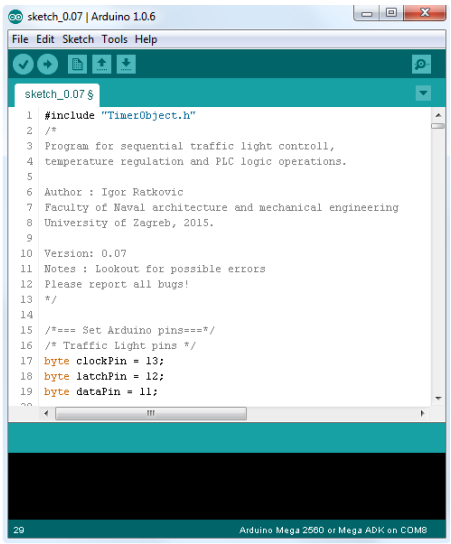
$$u(k) = u_{limit}(k) = u_P(k) + u_I(k) + u_D(k) \Rightarrow u_I(k) = u_{limit}(k) - u_{PD}(k) \tag{14}$$

4 Izrada programa za mikrokontroler

Izrada programa za mikrokontroler sastoji se od definiranja algoritma za svaki objekt regulacije. Program koji se izvodi mora se moći u svakom trenutku prekinuti ili promijeniti. ATmega328 ima tri tipa memorije :

- *Flash* memorija
- **SRAM** (engl. ***S****tatic* ***R****andom* ***A****ccess* ***M****emory*)
- **EEPROM** (engl. ***E****lectricaly* ***E****rasable* ***P****rogrammable* ***R****ead-****O****nly* ***M****emory*)

Program za upravljanje objekata regulacije nalazi se u *flash* memoriji ATmega328 integriranog kruga. *Flash* i **EEPROM** memorije su memorije za trajnu pohranu podataka, tj. čuvaju zapisane podatke kada je uređaj isključen. **SRAM** memorija je radna memorija mikrokontrolera koja služi za učitavanje programa i njegovo inicijaliziranje. Nakon prestanka rada mikrokontrolera svi podatci učitani u radnoj memoriji se brišu. Program za Arduino naziva se skica (engl. *sketch*). Sintaksa koja se koristi za izradu skice kombinacija je C/C++ programskog jezika uz ograničenja i posebnosti koje uvjetuje razvojno sučelje Arduino *Development Environment*. **ADE** (slika 21) je multiplatformsko razvojno sučelje koje omogućuje pisanje programa, traženje i uklanjanje pogrešaka (engl. *debugging*) i njegovo prevođenje u strojni jezik razumljiv Arduino mikrokontroleru. Svaki Arduino mikrokontroler dolazi s ugrađenim sustavom za pokretanje i izvršavanje strojnog koda - tzv. *bootloaderom*, koji omogućuje jednostavno prenošenje skica s računala u *flash* memoriju mikrokontrolera preko USB veze.



Slika 21: Arduino razvojno sučelje

Zbog ograničene memorije mikrokontrolera potrebno je pravilno deklarirati podatkovni vrstu varijabli sustava. Vrste podataka s njihovom duljinom riječi prikazani su u tablici 5

Tablica 5: Vrste podataka za Arduino Uno3

Naziv	Duljina	Primjer	Raspon	Opis
boolean	8 bit	true / false	0 ili 1	vrsta podataka za definiranje istintosti tvrdnje
char (character)	8 bit	'a'	$[-127, 127]$	vrsta podatka za označavanje slova i simbola u ASCII sustavu
unsigned char	8 bit	'ü'	$[0, 255]$	vrsta podatka za označavanje slova i simbola u proširenom ASCII sustavu
byte	8 bit	$B10010$	$[0, 255]$	vrsta podatka za pohranu broja ili bit operacije
int (integer)	16 bit	-1024	$[-32768, 32767]$	vrsta podatka za pohranu cjelobrojnog broja
unsigned int	16 bit	1024	$[0, 65535]$	vrsta podatka za pohranu cjelobrojnog pozitivnog broja
long	32 bit	-2^{10}	$[-2^{31}, 2^{31} - 1]$	vrsta podatka za pohranu velikih cjelobrojnih brojeva
unsigned long	32 bit	2^{20}	$[0, 2^{32} - 1]$	vrsta podatka za pohranu velikih pozitivnih cjelobrojnih brojeva
float / double	32/64 bit	$3,1459$	$[-3.4028235 \cdot 10^{38}, -3.4028235 \cdot 10^{38}]$	vrsta podatka za pohranu decimalnih brojeva, unutar Arduino Uno mikrokontrolera nema razlike između raspona float i double tipa podatka [16]

4.1 Osnovna struktura programa

Program je koncipiran kao konačni automat (engl. *finite state machine*), u svakom trenutku program se nalazi u jednom stanju i prebacuje se u drugo stanje tek nakon što se zadovolji uvjet prelaska u sljedeće stanje. Uvjet prebacivanja iz jednog stanja u drugo je primanje naredbe putem serijske veze za drugom programskom rutinom ili reinicijalizacija trenutne rutine s novim podatcima. Osnovna struktura programa može se podijeliti u četiri segmenta:

- upotreba vanjskih programskih paketa i definiranje globalnih varijabli
- definiranje početnog stanja u funkciji *setup()*
- smještanje programa unutar nadređene petlje *loop()*
- funkcije izvan petlje koje se pozivaju po potrebi

koji su prikazani na primjeru koda 1

```
# include custom_library.h // programski paketi
/* 1. Odredivanje globalnih vrijednosti */
```

```
int pin = 10;
char b[2];
/* 2. Postavljanje prikljucaka i pocetak serijske komunikacije */
void setup()
{
  pinMode(pin, OUTPUT);
  Serial.begin(115200);
}
/* 3. Glavna petlja programa */
void loop()
{
  /* sadrzi sva stanja programa
}
/* 4. Definiranje funkcija van glavne petlje */
void doSomething()
{
}
```

Kod 1: Osnovna struktura Arduino programa

Glavna stanja u kojima se program može naći su sljedeća:

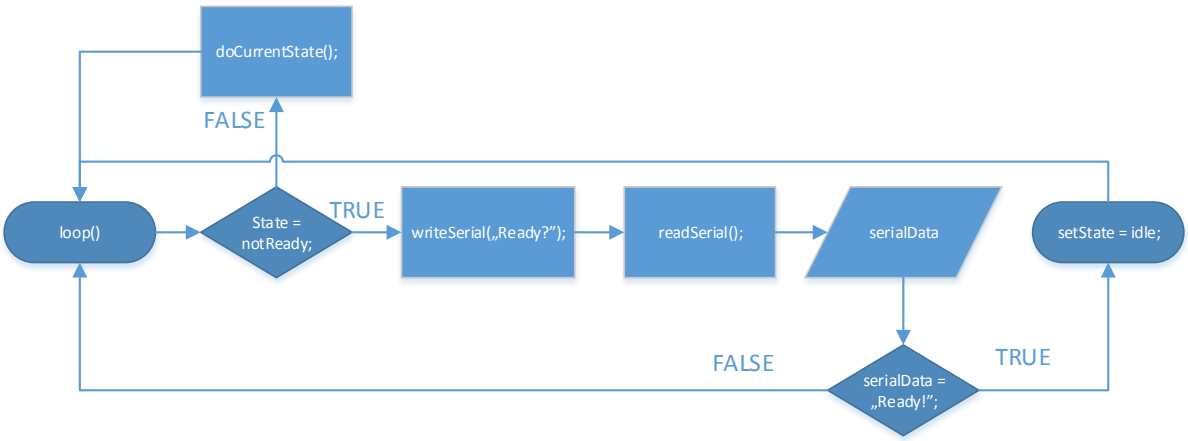
- iščekivanje početka serijske komunikacije
- čitanje i obrada podataka iz serijskog spremnika
- sekvencijalno upravljanje semaforima
- simuliranje osnovnog rada PLC-a
- praćenje stanja toplinske komore
- okidna regulacija temperature toplinske komore
- PID regulacija temperature toplinske komore
- stanje mirovanja

Prijelaz iz jednog glavnog stanja u drugo postiže se promjenom upravljačke varijable (engl. *controll expression*) *switch* naredbe. Primjer primjene *switch* logike (logike diskretnih stanja) prikazan je u segmentu koda 2.

```
void loop()
{
  switch(controllExpression)
  {
    case State1:
      doState1();
    break;
    case State2:
      doState2();
    break;
    default:
      doState3();
  }
}
```

Kod 2: Primjer promjene stanja upotrebom *switch* naredbe

Nakon postavljanja i inicijaliziranja globalnih varijabli i stanja priključaka na mikrokontroleru, program ulazi u beskonačnu petlju. Slika 22 prikazuje tijek izvršavanja koji se ponavlja sve do trenutka kada nadređeni kontroler ostvari serijsku komunikaciju i pošalje odgovarajuću potvrdu o početku komunikacije.



Slika 22: Početno stanje mikrokontrolera

Arduino također ima bogatu biblioteku programskih paketa (skupova specifičnih potprograma), koji omogućavaju jednostavno proširenje funkcionalnosti programa pozivanjem paketa u zaglavlju skice. Jedini korišteni programski paket je *Arduino Timer Object* [17]. *Timer Object*

omogućuje stvaranja brojila vremena (engl. *timer*) s funkcijom povratnog poziva (engl. *callback function*). Funkcija povratnog poziva najčešće je izvršna funkcija glavnog stanja u kojem se program nalazi. Brojila vremena koriste se za zadržavanje rezultata neke funkcije dovoljno dugo da korisnik to vidi (npr. držanje svjetlosne diode u visokom stanje) ili ako je potrebno pozivati funkciju periodički s proizvoljno određenim vremenskim intervalom.

4.2 Serijska komunikacija

Početak serijske komunikacije unutar mikrokontrolera definiran je unutar funkcije *setup()* pozivanjem funkcije

```
void setup()
}
  Serial.begin(baudRate);
}
```

Kod 3: Postavljanje početka serijske komunikacije

gdje je varijabla *baudRate* brzina prijenosa podataka u baudima. Brzina prijenosa podataka između podređene i nadređene jedinice mora biti identična, inače dolazi do nesuvisle komunikacije. Dolazni podatci se spremaju u dolazni spremnik (engl. *buffer*) veličine $64Byte$. Ostale postavke serijske komunikacije su:

- duljina podatkovne riječi je jedan bajt, odnosno osam bitova
- jedan start i stop bit
- ne koristi se provjera pariteta.

Serijska komunikacija između dva uređaja je postupak slanja jednog po jednog bita, sekvencijalno, preko istog komunikacijskog kanala. Kako bi se osiguralo da podređeni mikrokontroler prima ispravne podatke za upravljanje određenim objektom potrebno je odrediti komunikacijski protokol. Komunikacijski protokol je jasno definirani sustav pravila za izmjenu podataka između dva uređaja. Pravila koje poruka slana s nadređenog uređaja mora zadovoljiti ovise o programu koji se želi izvršavati. Ono što je zajedničko svakoj poruci je postojanje zaglavlja i podnožja. Zaglavlje definira početak primanja poruke. U slučaju da zaglavlje nije primljeno program će zanemariti sve podatke dok ne očita podatak koji definira zaglavlje programa. Podnožje programa je terminirajući podatak serijske komunikacije nakon kojeg se zaprimljeni

podatci obrađuju i pohranjuju za korištenje unutar glavnih stanja.

Opći zapis protokola sastoji se od šest elemenata :

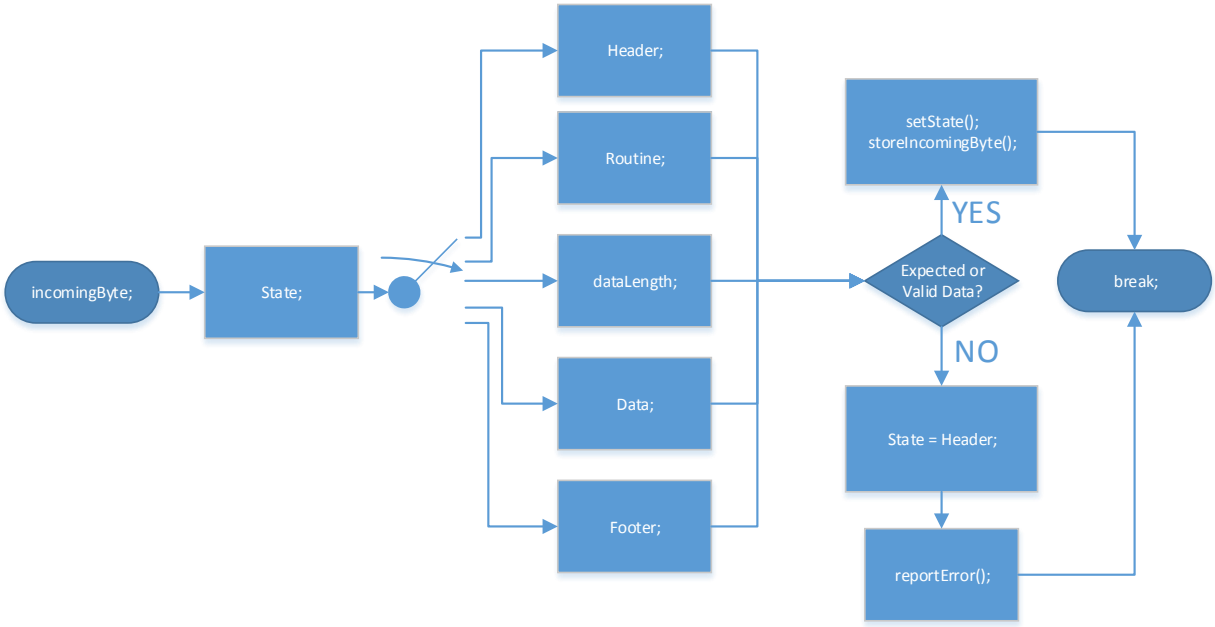
$$< \quad \textbf{rutina} \quad \textbf{N} \quad : \quad \textbf{podatci} \quad > .$$

Tablica 6 prikazuje elemente protokola i njihovu ulogu u protokolu.

Tablica 6: Elementi komunikacijskog protokola

Element protokola	**Opis**	**Vrsta podataka**	**Dodatni članovi**
<	Zaglavlje poruke, označava početak poruke		
rutina	Rutina poruke određuje program	niz slova (engl. *string*)	
N	Duljina podataka za program određen rutinom	cjelobrojni broj	
:	Uloga graničnika, označava kraj duljine podataka i početak primanja podataka		
podatci	Svi podatci koji su potrebni programu za izvršavanje	ovisi o rutini, mogu biti cijeli brojevi, decimalni ili slova	točka i zarez kao graničnici između niza podataka
>	Podnožje poruke, označava kraj poruke		

Čitanje i obrada podataka primljenih serijskom vezom može se gledati kao niz stanja u kojemu sljedeće stanje ovisi o prethodno obrađenom podatku i trenutnom podatku. Slika 23 prikazuje tijek čitanja podataka sa serijske veze. Prilikom svake iteracije glavne petlje, mikrokontroler provjerava postoji li podatak u serijskom spremniku. Podatak iz serijskog spremnika proslijeđuje se funkciji koja vrši obradu trenutnog podatka. Ako je podatak ispravan on se sprema u odgovarajući globalni spremnik, no ako je podatak neispravan mikrokontroler javlja grešku i vraća se u početno stanje. Trenutni podatak određuje moguću promjenu stanja koja će obraditi sljedeći podatak u serijskom spremniku. To omogućava preskakanje određenih stanja ovisno o prepoznatoj vrsti poruke.



Slika 23: Čitanje i obrada podataka primljenih serijskom vezom

4.3 Izrada programa za sekvencijalno upravljanje semafora

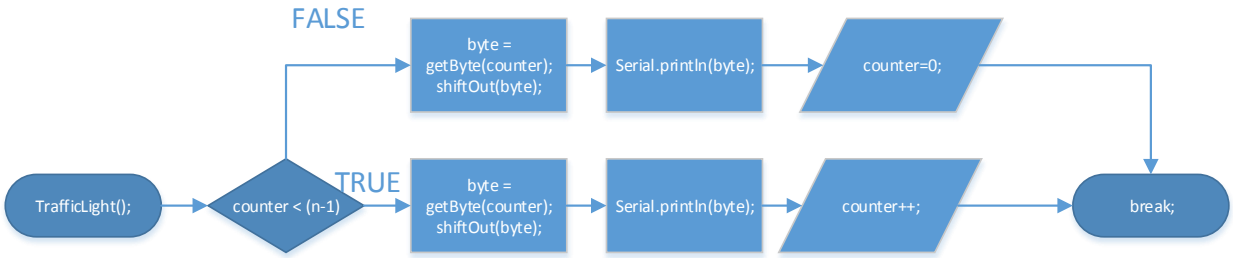
Iz analize sekvencijalnog rada semafora vidljivo je da je za njegovo upravljanje dovoljan jedan bajt. Drugim riječima, sve moguće kombinacije semafora opisane su vrijednostima od $0 - 255$. Zbog kraćeg zapisa, umjesto binarnog odabran je dekadski zapis vrijednosti stanja semafora. Rutina za označavnje semafora glasi :

$$\textbf{rutina} = "TL"$$

gdje je *"TL"* akronim od engleskog izraza za semafor, *Traffic Light*. Pravilna poruka za sekvencijalno upravljanje s duljinom podataka n ima sljedeći oblik:

$$< TLn : b_1, b_2, b_3, \cdots, b_n >, \quad n \in \mathbb{N}_1,\ b \in \mathbb{N}_0 : n < 256$$

Postupak sekvencijalnog upravljanje semaforom prikazan je na slici 24:
Funkcija za izvršavanje sekvencijalnog upravljanja poziva se svake dvije sekunde. To je vremenski period koji je dovoljan da korisnik usporedi poslane podatke s rezultatom upravljanja na fizičkoj maketi. Za dodatnu kontrolu, Arduino svaki korak šalje trenutnu vrijednost nadređenom regulatoru.



Slika 24: Algoritam sekvencijalnog upravljanja

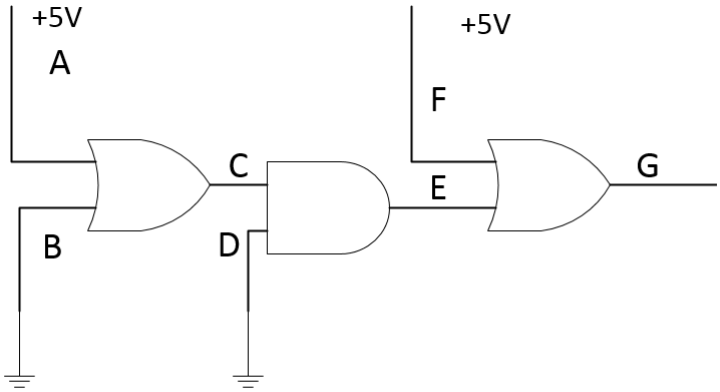
4.4 Izrada programa za PLC

Izvođenje logičkih operacija provodi se upotrebom logičkih (Booleovih) operatora [18] prikazanih u tablici 7. Podatci potrebi za simulaciju temeljnog rada PLC-a su oznake logičke operacije

Tablica 7: Logičke operacije i operatori

Logička operacija	**Simbol operatora**
I	&&
ILI	||
NE	!

i priključci na koji se te operacije odnose. Karakteristika logičkih operacija je da mogu imati više ulaza ali samo jedan izlaz. Izlaz jedne logičke operacije može biti ulaz druge kao što je prikazano na slici 25.



A | B		C	C & D		E	E | F		G
1	0	1	1	0	0	0	1	1

Slika 25: Serijski spoj logičkih operatora i tablica istinitosti

Podatci se obrađuju serijski, tako da se za sljedeću logičku operaciju uzima rezultat prethodne operacije i stanje priključka na koji se ta operacija odnosi. Za rješavanje ulančanih logičkih operacija potrebno je uvesti pomoćnu varijablu A_{cc} koja će pohraniti rezultat prethodne operacije i služiti kao ulaz sljedećoj logičkoj operaciji.

```
/* int A, F = 1; */
/* int B, D = 0; */
int Acc;
Acc = (A || B);
Acc = (Acc && D);
Acc = (Acc || F);
```

Kod 4: Serijski spoj logičkih operatora

Osim logičkih operacija uvode se i operacije inicijalizacije pomoćne varijable i spremanja stanja pomoćne varijable na izlaz. Spremanje stanja pomoćne varijable je postavljanje priključka mikrokontrolera na vrijednost pomoćne varijable. Inicijalizacija pomoćne varijable je postavljanje vrijednosti pomoćne varijable na vrijednost logičkog ulaza ili izlaza priključka mikrokontrolera. Popis svih operacija za simuliranje temeljnog rada PLC-a i njihov opis prikazan je tablicom 8, dok tablica 9 prikazuje na koju vrijednost se odnosi koji ulazno/izlazni priključak.

Tablica 8: Popis operacija za simuliranje rada PLC-a

Oznaka operacije/priključak (*pin*)	**Logička operacija**	**Opis**
Lpin	Acc = digitalRead(pin)	Dodjeluje očitanu vrijednost pomoćnoj varijabli
LCpin	Acc = !(digitalRead(pin))	Dodjeluje komplementarnu očitanu vrijednost pomoćnoj varijabli
Spin	digitalWrite(pin, Acc)	Postavlja pin u stanje pomoćne varijable
SCpin	digitalWrite(pin, !(Acc))	Postavlja pin u komplementarno stanje pomćne varijable
ORpin	Acc = Acc —— digitalRead(pin)	Operacija ILI
ORCpin	Acc = Acc —— !(digitalRead(pin))	Komplementarno ILI
ANDpin	Acc = Acc && (digitalRead(pin)	Operacija I
ANDCpin	Acc = Acc && !(digitalRead(pin))	Komplementarno I
NOTpin	Acc = !(Acc)	Operacija NE
XORpin	Acc = Acc ^digitalRead(pin)	Ekskluzivno ILI

Rutina za označavanje PLC-a glasi :

$$\textbf{rutina} = "PLC".$$

Tablica 9: Mapiranje vrijednosti priključaka u poruci s fizičkim ulazima/izlazima

	Ulazi				**Izlazi**			
Fizički priključak	In1	In2	In3	In4	Out1	Out2	Out3	Out4
Vrijednost priključka za poruku	1	2	3	4	11	12	13	14

Pravilna poruka za simuliranje rada PLC-a duljine podataka n ima sljedeći oblik:

$$< PLCn : a_0, a_2, a_3, \cdots, a_n >, \quad n \in \mathbb{N}_1$$

gdje je a kombinacija logičke operacija i priključka na kojeg se odnosi, npr. :

$$a = AND13$$

primjenuje logičku operaciju **I** na pomoćnu varijablu A_{cc} i na vrijednost trećeg izlaznog priključka. Prilikom rada PLC program ne ispisuje vrijednosti nadređenom kontroleru već se rezultat logičkih operacija očituje na fizičkom modelu makete.

4.5 Izrada programa za automatsku regulaciju toplinskog procesa

U poglavlju 4.1 određeno je da se automatska regulacija toplinskog procesa sastoji od tri glavna stanja programa :

- praćenje stanja toplinske komore (otvoreni regulacijski krug)
- relejna dvopoložajna regulacija temperature toplinske komore
- PID regulacija temperature toplinske komore.

Pripadajuća funkcija glavnog stanja poziva se svaku sekundu, iz čega proizlazi da je vrijeme uzorkovanja $T_0 = 1\,s$. Praćenje stanja toplinske komore je proces okidanja tranzistora snage i držanja grijaćeg tijela toplinske komore u zatvorenom strujnom krugu, a također služi kao statusni bit za snimanje prijelazne karakteristike sustava u otvorenom regulacijskom krugu. Praćenje stanja ne zahtijeva dodatne podatke od korisnika a rutina za ovaj program glasi:

$$\textbf{rutina} = "TC" \tag{15}$$

gdje je "TC" akronim od engleskog izraza za toplinsku komoru, *Thermal Chamber*. Pravilna poruka za praćenje stanja toplinske komore ima sljedeći oblik:

$$< TC > .$$

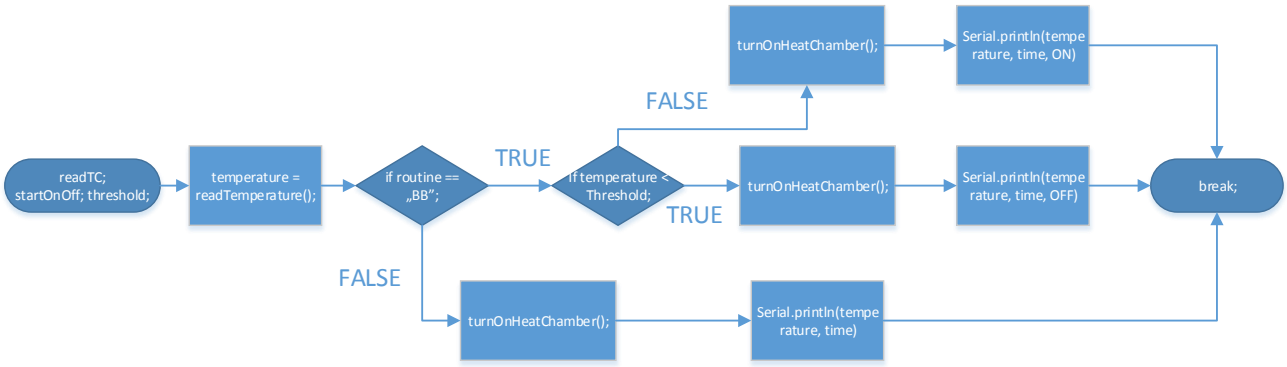
Relejna dvopoložajna regulacija toplinske komore zahtijeva od korisnika određivanje referentne temperature (ϑ). Rutina za relejnu dvopoložajnu regulaciju glasi:

$$rutina = "BB",$$

gdje je "BB" akronim od engleskog izraza za relejna dvopoložajnu regulaciju, *Bang-Bang*. Pravilna poruka za okidno upravljanje temperaturom toplinske komore:

$$< BB1 : \vartheta >, \quad \vartheta \in \mathbb{N}$$

Relejna regulacija temperature toplinske komore može se promatrati kao proširenje programa za praćenje stanja. Jedina razlika je ta što ova vrsta regulacije zahtijeva od korisnika definiranje temperaturne granice okidanja (ϑ).



Slika 26: Dijagram toka funkcije praćenja stanja i relejne regulacije toplinske komore

Algoritam za PID regulaciju određen je jednadžbom 12. Za PID regulaciju korisnik mora odrediti referentnu temperaturu i iznose pojačanja K_p, K_i, K_d. Rutina za ovaj program glasi:

$$\textbf{rutina} = "PID".$$

Pravilna poruka za PID regulaciju temperature toplinske komore ima sljedeći oblik:

$$< PIDn : \vartheta, K_p, K_i, K_d >, \quad \vartheta \in \mathbb{N}; K_p, K_i, K_d \in \mathbb{Q} : K_p, K_i, K_d \geq 0$$

Kod 5 prikazuje implementaciju PID regulatora u programskom obliku.

```
double samplingTime = 1;
double uI = 0;
double oldTheta;
double oldError;

void runPID()
{
  double referenceValue = processControllerData[0];
  double Kp = processControllerData[1];
  double Ki = processControllerData[2];
  double Kd = processControllerData[3];
  int analogValue =  analogRead(temperaturePin);
  double theta = voltageReference*(analogValue/1024.0)*100;
  double error = referenceValue - theta;
  double uP = Kp * error;
  double uD = (error - oldError) / samplingTime;
  uI += Ki * samplingTime * error;
  double output = uP + uI + uD;
/* Limitiranje izlaza i antiwindup reset integratora */
  if(output > 255){
    output = 255;
    uI = output - uP - uD;}
  if(output <= 0){
    output = 0;
    uI = -uP - uD;}
/* Postavi radni ciklus na PWM prikljucku */
  analogWrite(regulateTC, output);
  oldTheta = theta;
  oldError = error;
}
```

Kod 5: Programski oblik PID regulatora

4.6 Ponovno postavljanje stanja

Posljednje stanje u kojem se program može nalaziti je ujedno i prvo stanje u koje program ulazi nakon potvrđenog ostvarivanja komunikacije. Mikrokontroler se nalazi u stanju mirovanja (engl. *idle state*) i ostaje u tom stanju sve dok odgovarajuća poruka ne stigne od strane korisnika. Korisnik u bilo kojem trenutku može izaći iz trenutnog stanja i ući u stanje mirovanja, prilikom čega dolazi do ponovnog postavljanja varijabli i priključaka na početne vrijednosti. Rutina za potvrdu početka komunikacije ili ponovnog postavljanja je identična:

$$\textbf{rutina} = "R" \tag{16}$$

gdje "R" dolazi od prvog slova engleskog izraza za spremnost (engl. *Ready*), ali i od od prvog slova engleskog izraza za ponovno postavljanje stanja (engl. *Reset*). Pravilna poruka za početak komunikacije ili za ponovno postavljanje glasi:

$$< R > .$$

Tablica 10 prikazuje sve pravilne poruke za inicijalizaciju određenog stanja programa unutar mikrokontrolera.

Tablica 10: Pregled svih poruka za odabir stanja programa mikrokontrolera

Stanje programa	**Rutina**	**Primjer poruke**	**Izlazni podatci programa u tijeku**
stanje mirovanja, iščekivanje početka serijske komunikacije	R	<R>	"Reset!"
sekvencijalno upravljanje semaforima	TL	<TL5:255,128,64,32,16>	trenutna dekadska vrijednost sekvencijalnog upravljanja
simuliranje osnovnog rada PLC-a	PLC	<PLC5:L1,OR12,SC11,OR2,SC12>	
praćenje stanja toplinske komore	TC	<TC>	vrijeme i temperatura
okidna regulacija temperature toplinske komore	BB	<BB1:30>	vrijeme, temperatura i stanje grijanja
PID regulacija temperature toplinske komore	PID	<PID4:30,5.2,2,0.5>	vrijeme, temperatura, regulacijsko odstupanje i vrijednost upravljačke varijable

5 Izrada grafičkog korisničkog sučelja

Grafičko korisničko sučelje (engl. **GUI**, ***G**raphical **U**ser **I**nterface*) omogućava korisniku interakciju s računalom pomoću aktivnih grafičkih elemenata. Grafičko sučelje predstavlja značajan odmak od sučelja temeljenih na tekstualnom prikazu i unosu. Rad unutar grafičkog sučelja je intuitivniji i ne zahtjeva prethodno poznavanje tekstualnih naredbi. Grafički element (engl. *widget*) je osnovna građevna jedinica grafičkog sučelja.
Za izradu korisničkog sučelja korišten je Python programski jezik. Svi korišteni programi i njihovi programski paketi funkcija otvorenog su koda i slobodni za korištenje u akademske svrhe. Korišteni programski alati :

- Python 2.7.3 [19]
  - GTK+3.10.2 [23]
  - NumPy 1.6.2 [25]
  - matplotlib 1.4.3 [26]
  - pySerial 2.7 [29]

Navedena su samo osnovna proširenja, koja za pravilno izvršavanje zahtijevaju instalaciju dodatnih programskih paketa funkcija (engl. *dependancies*). Paketi se preuzimaju s Raspbian repozitorija [20] upisivanjem odgovarajuće naredbe u terminal Raspberry Pi-a:

$$sudo \quad apt - get \quad python -' imepaketa'.$$

Zbog opširnosti koda, princip izrade grafičkog korisničkog sučelja bit će objašnjen na temelju jednostavnih primjera. Kod grafičkog sučelja za simulaciju i upravljanje rada edukacijske makete dan je u digitalnom obliku na DVD-u u prilogu diplomskog rada.

5.1 Python

Python je interpreterski, interaktivni i objektno orijentirani programski jezik, kojeg je 1990. godine razvio Guido von Rossum. Neke od značajnih odlika Python programskog jezika su čista i logična sintaksa, interpretacija međukoda, proširivost, multiplatformska podrška i otvorenost koda. Python olakšava pisanje programa i omogućuje korisniku da se više usredotoči na problem koji rješava nego na samo pisanje programa. Python ima i snažnu podršku mnogobrojne

zajednice okupljenu oko *Python Software Foundation*, neprofitne organizacije s ciljem promicanja i stalnog unaprijeđivanja tog programskog jezika. Zahvaljujući principu otvorenog koda (eng. *open source*), Python je moguće proširivati brojnim modulima, odnosno programskim paketima stvorenima od strane Python zajednice koji znatno olakšavaju realizaciju rješenja zamišljenog problema.

```
import os
import platform
from gi.repository import Gtk, GObject, Gdk
import serial
import numpy as np
import matplotlib.pyplot as plt
```

Kod 6: Učitavanje potrebnih programskih paketa

5.2 GTK+ [24]

GTK+ je knjižnica funkcija koja služi za izradu grafičkog sučelja, i sadrži brojne grafičke elemente, poput gumba, padajućih izbornika, spremnika teksta i drugo. GTK+ je kompatibilan s brojnim operacijskim sustavima zasnovanima na UNIX-u (npr. Raspbian), kao i s Windows i OS X operacijskim sustavima. Gdk je modul koji služi kao posrednik između korisničkog sučelja i operacijskog sustava i omogućava detektiranje događaja unutar sučelja. GObject je modul unutar GTK+ koji služi za stvaranje grafičkih elemenata, obrađivanje raznih vrsta podataka i upravljanje događajima i signalima.

GTK+ je sustav upravljan događajima. Program se odvija unutar glavne programske petlje koja stalno provjerava prisutnost novog događaja, i na temelju događaja emitira signal koji je moguće povezati s grafičkim elementom. Događaji su sve promjene od strane korisnika preko ulaznih jedinica, kao što su miš i tipkovnica. Glavni objekt za upravljanje grafičkim elementima, iz kojeg naslijeđuju svi ostali grafički elementi je *Gtk.Widget*. *Gtk.Widget* je zadužen za životni ciklus podređenog objekta, njegovo stanje, veličinu i izgled. GTK+ korisničko sučelje tvori se od smještanja jednog grafičkog elementa unutar drugog.

Ovisno o broju elemenata koje spremnik može sadržavati, spremnike (engl. *container*) dijelimo na :

- spremnike za jedan element - *Gtk.Bin*
- spremnike za više elemenata - *Gtk.Box*, *Gtk.Grid*.

Spremnici za jedan element određuju vizualna i interakcijska svojstva podređenog grafičkog elementa koji je u njima sadržan. Primjeri takvih grafičkih elemenata su:

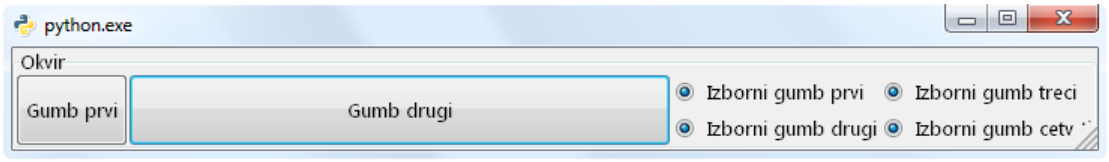
- *Gtk.Window* - glavni prozor programa
- *Gtk.Button* - gumb kojeg je moguće kliknuti
- *Gtk.Frame* - okvir oko podređenog objekta

Spremnici za više elemenata služe za upravljanje prostornim rasporedom podređenih grafičkih elemenata. Spremnici određuju veličinu i raspored podređenih grafičkih elemenata. Redoslijed spremanja elemenata u spremnike ovisi o vrsti spremnika. Tako se unutar *Gtk.Box* spremnika elementi spremaju u vektor stupac ili u vektor redak, dok se unutar *Gtk.Grid* spremnika elementi spremaju u matricu.

```
from gi.repository import Gtk
# -- Stvaranje grafickih elemenata -- #
window = Gtk.Window()
frame = Gtk.Frame()
frame.set_label('Okvir')
box = Gtk.Box()
button0 = Gtk.Button('Gumb prvi')
button1 = Gtk.Button('Gumb drugi')
grid = Gtk.Grid()
radio0 = Gtk.RadioButton('Izborni gumb prvi')
radio1 = Gtk.RadioButton('Izborni gumb drugi')
radio2 = Gtk.RadioButton('Izborni gumb treci')
radio3 = Gtk.RadioButton('Izborni gumb cetvrti')
#-- Spremanje grafickih elemenata u spremnike -- #
window.add(frame)
frame.add(box)
box.pack_start(button0, False, False, 0)
box.pack_start(button1, True, True, 0)
box.pack_start(grid, False, False, 0)
grid.attach(radio0, 1, 1, 1, 1)
grid.attach(radio1, 1, 2, 1, 1)
grid.attach(radio2, 2, 1, 1, 1)
grid.attach(radio3, 2, 2, 1, 1)
#-- Prikaz -- #
window.show_all()
```

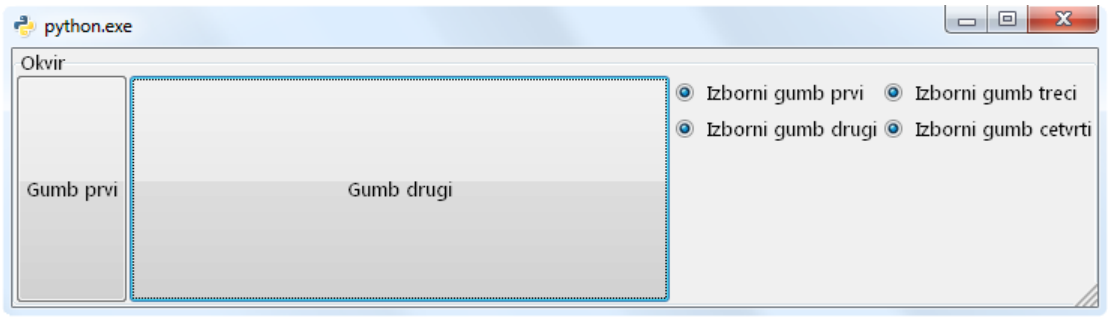
```
#-- Glavna programska petlja sucelja --#
Gtk.main()
```

Kod 7: Primjer stvaranja i smještanja grafičkih elemenata u spremnike



Slika 27: Grafičko sučelje iz primjera koda 7

Prilikom spremanja grafičkog elementa u nadređeni spremnik, moguće je definirati koliko prostora element može zauzeti. Ovisno o postavkama, element može zauzeti samo najnužniji prostor potreban za prikaz ili može popuniti sav prostor koji spremnik sadrži. Grafički element uvijek zauzima sav raspoloživi prostor u vertikalnom smjeru ako se nalazi u vektoru redku (slika 28), odnosno sav raspoloživi prostor u horizontalnom smjeru ako se nalazi u vektor stupcu.



Slika 28: Grafičko sučelje iz primjera koda 7, povećanjem početne veličine prozora dolazi do promjene veličina elemenata unutar vektorskih spremnika

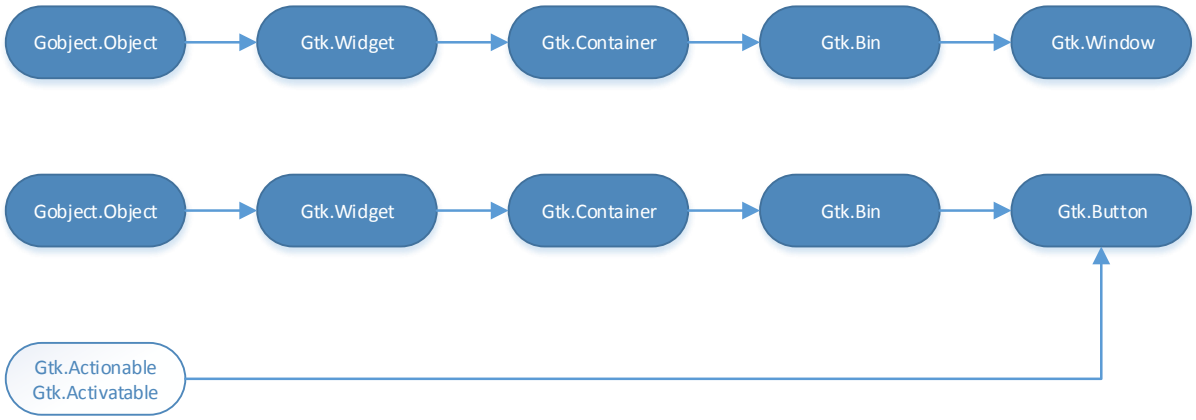
```
#-- Povezivanje grafickih elemenata sa signalima --#
button0.connect('clicked', onClickDoSomething)
radio0.connect('toggled', onToggleChangeSomething)

def onClickDoSomething(widget):
    #-- Akcija/radnja vezana uz graficki element -- #
    pass

def onToggleChangeSomethin(widget):
    #-- Akcija/radnja vezana uz graficki element -- #
    pass
```

Kod 8: Primjer povezivanja grafičkih elemenata sa signalima

Grafički elementi sa slike 27 su neinteraktivni. Pritiskom na gumb ne dolazi po vidljivih promjena ili promjena u pozadini sučelja. Kako bi se grafički element učinio interaktivnim potrebno je povezati odgovarajući signal s grafičkim elementom i odrediti funkciju povratnog poziva koja se provodi u slučaju pojave tog signala (kod 8). Pojedini grafički elementi mogu se povezati s točno određenim signalima. Ostali elementi zahtijevaju korištenje dodatnog nadređenog spremnika kojeg je moguće povezati s proizvoljnim signalom. Slika 29 prikazuje razliku između prozora sučelja (*Gtk.Window*) i gumba (*Gtk.Button*). Oba grafička elementa su aktivnog tipa, s razlikom da prozor sučelja nije moguće spojiti s proizvoljnom funkcijom povratnog poziva, jer element ne nasljeđuje to svojstvo.



Slika 29: Nasljeđivanje svojstva grafičkog elementa

5.3 matplotlib [27]

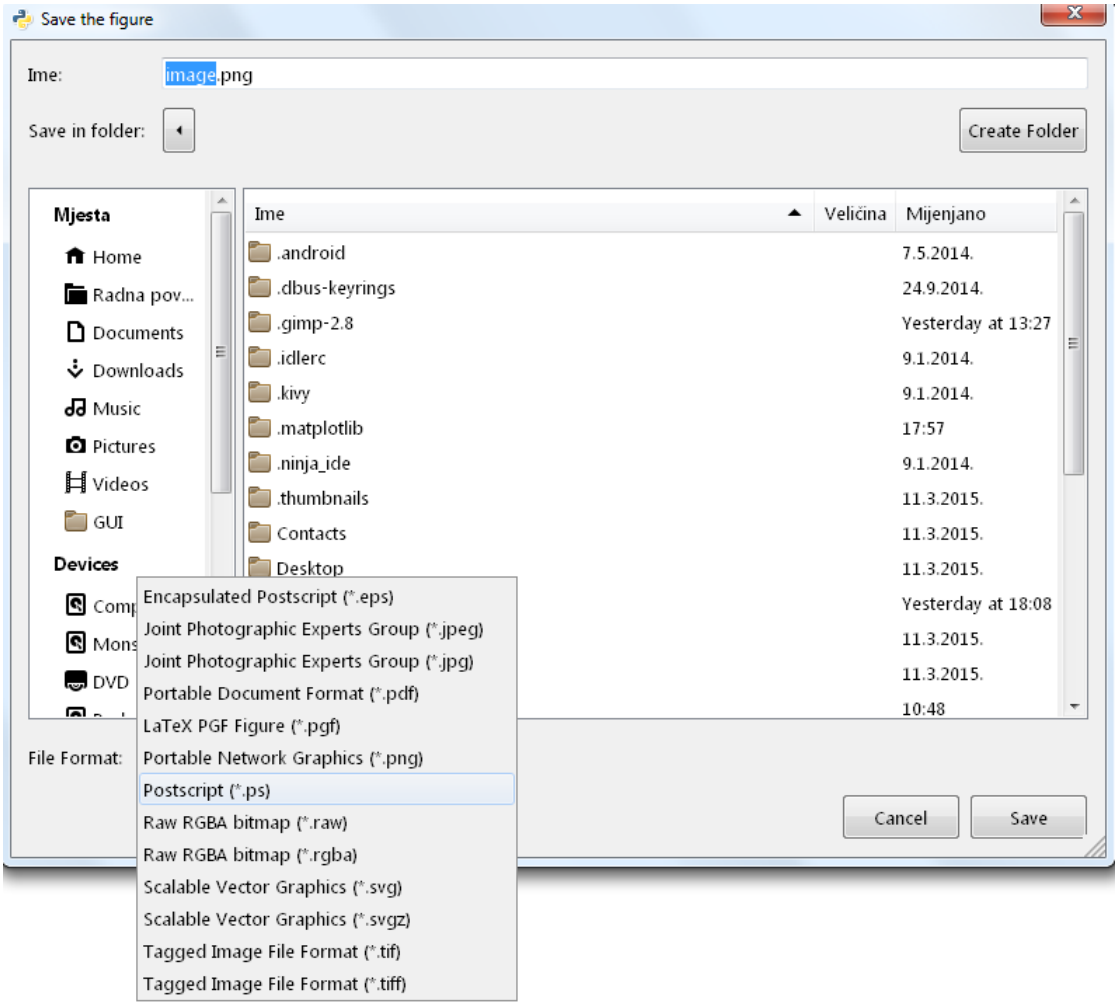
Matplotlib je programski paket funkcija koje služe za iscrtavanje crteža visoke kvalitete. Omogućuje neinteraktivno i interaktivno iscrtavanje i spremanje slika u različitim formatima (slika 30). Standardni pozadinski sustav za prikaz slike je **Agg** (engl. ***A**nti-**G**rain **G**eometry*), koji služi za prikaz slike vrlo visoke kvalitete, s funkcijom zaglađivanja rubova (engl. *antialiasing*) i podpikselnom rezolucijom. Za prikaz slike (engl. *render*) unutar grafičkog sučelja (engl. *embedded*) potrebno je koristiti odgovarajući pozadinski sustav (engl. *back-end*) za prikaz slike. Pozadinski sustav za prikaz sastoji se od dva strukturirana sloja :

- sloj za iscrtavanje i prikaz slike
- sloj za smještanje slike (engl. *canvas*)

Za korištenje matplotlib grafova unutar GTK+3 grafičkog sučelja potrebno je koristiti **GTK3Agg** pozadinski sustav za prikaz.

```
import matplotlib
matplotlib.use('GTK3Agg')
from matplotlib.figure import Figure
import matplotlib.animation as animation
from matplotlib.backends.backend_gtk3agg import FigureCanvasGTK3Agg as
    FigureCanvas
from matplotlib.backends.backend_gtk3 import NavigationToolbar2GTK3 as
    NavigationToolbar
```

Kod 9: Učitavanje potrebnih matplotlib paketa za ugrađeni prikaz grafova



Slika 30: Različiti formati za spremanje grafova

5.4 PySerial [30]

PySerial je programski paket koji sadrži funkcije potrebne za ostvarivanje serijske komunikacije između nadređenog kontrolera i podređenog mikrokontrolera. Nakon ostvarivanja komunikacije moguće je primati i slati poruke. Serijska komunikacija započinje inicijalizacijom objekta s odgovarajućim parametrima prikazanim kodom 10.

```
import serial
openConnection = serial.Serial(port, baudrate, bytesize, timeout)
```

Kod 10: Ostvarivanje serijske komunikacije

gdje je:

- *port* - ime priključka
- *baudrate* - brzina prijenosa podataka u baudima, mora odgovarati brzini prijenosa određenog kod mikrokontrolera
- *bytesize* - duljina podatkovne riječi, odabire se duljina od osam bitova
- *timeout* - vremenski interval čitanja podataka serijske veze.

Ime priključka ovisi o operacijskom sustavu na kojem se program izvodi. Ako je riječ o operacijskom sustavu baziranom na Linux jezgri, ime priključka imat će formu :

$$port = '/dev/ttyACM\mathbf{n}', \quad \mathbf{n} \in \mathbb{N}.$$

Odnosno ako je riječ o Windows operacijskom sustavu:

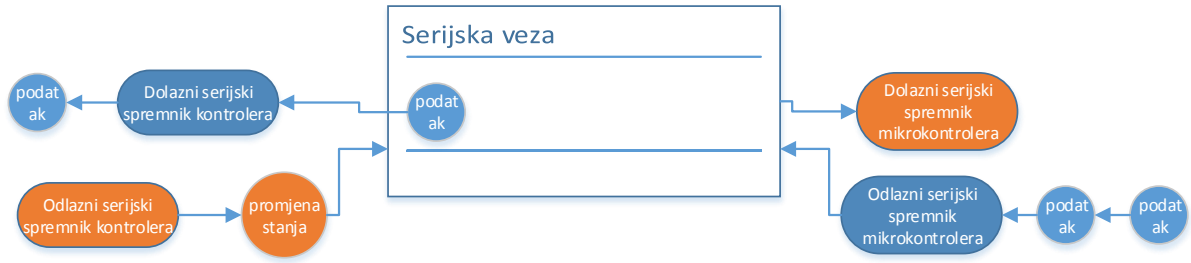
$$port = '\backslash\backslash.\backslash COM\mathbf{n}', \quad \mathbf{n} \in \mathbb{N}.$$

Primanje i slanje podataka preko serijske veze prikazano je primjerom 11.

```
import serial
#-- Podatci za slanje moraju biti u obliku niza slova (engl. string) --#
data = '<TL3:64,127,255>'
#-- Slanje podataka serijskom vezom --#
serial.Serial.write(data)
#-- Primanje podataka serijskom vezom,
# podatci se citaju iz serijskog spremnika sve
```

```
# dok se ne naide na bajt koji oznacava novi red '\n' --#
incomingData = serial.Serial.readline()
```

Kod 11: Primanje i slanje podataka preko serijske veze



Slika 31: Ilustracija serijske komunikacije između nadređenog i podređenog kontrolera

Slika 31 prikazuje primanje podataka s podređenog mikrokontrolera u trenutku slanja poruke za promjenu programa upravljanja na mikrokontroleru. Nadređeni kontroler primit će poruku koja se trenutno šalje i spremiti ju u svoj dolazni spremnik. Nakon toga nadređeni kontroler šalje poruku za promjenu stanja programa mikrokontrolera. Poruka će se spremiti u dolazni spremnik mikrokontrolera i obrađivati bajt po bajt. Svaki bajt poruke obrađuje se u jednom ciklusu glavne petlje mikrokontrolera. Dok traje obrada primljene poruke, mikrokontroler se i dalje nalazi u starom stanju i u svakom ciklusu provodi upravljanje objekta, te rezultate upravljanja šalje serijskom vezom. To traje sve dok se primljena poruka ne obradi i ne promijeni stanje programa mikrokontrolera. Kako bi se izbjegla obrada pogrešnih podataka potrebno je isprazniti (engl. *flush*) spremnike serijske veze nadređenog kontrolera nakon svake naredbe za promjenom stanja programa mikrokontrolera (kod 12).

```
import serial
def flushSerial():
#-- Isprazni odlazni spremnik --#
    serial.Serial.flushOutput()
#--Isprazni dolazni spremnik --#
    serial.Serial.flushInput()
```

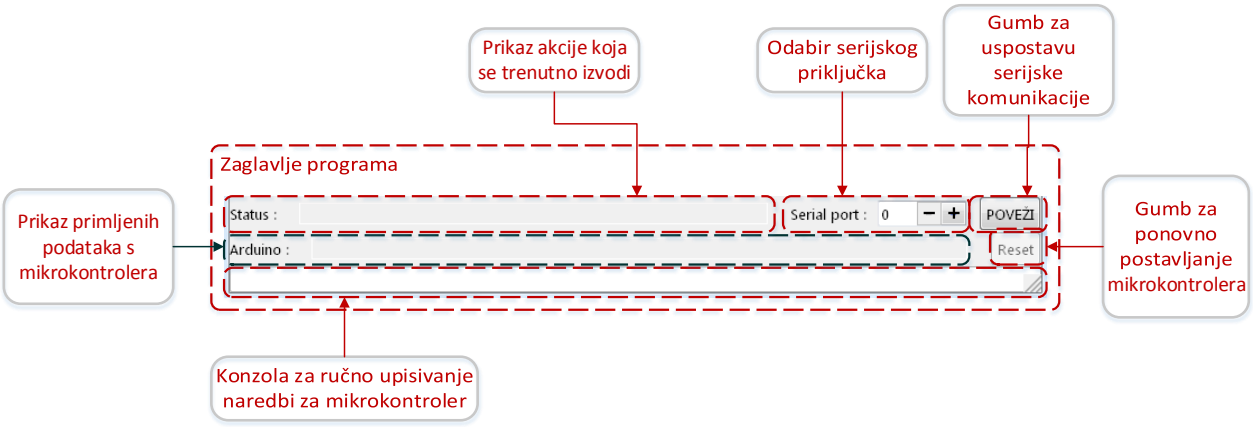
Kod 12: Pražnjenje odlaznog i dolaznog spremnika serijske veze nadređenog kontrolera

5.5 Struktura grafičkog sučelja

Grafičko sučelje sastoji se od tri kartice koje sadrže grafičke elemente za prikaz i simulaciju rada objekta upravljanja edukacijske makete. Uz kartice programa prisutno je i zaglavlje programa koje je zajedničko svim programskim segmentima. Programska struktura grafičkog sučelja prikazana je na primjeru koda 13.

```
#-- Potrebni programski paketi --#
import ...
#-- Glavna programska klasa --#
class Main:
    def __init__(self):
  # -- Inicijalizira klase za stvaranje sucelja i povezivanje
  # elemenata sucelja s odgovarajucim signalima,
  # prikazuje sve elemente sucelja i
  # pokrece glavnu petlju programa --#
class MainWindow:
  # -- Stvara sve graficke elemente
  # i sprema ih u odgovarajuce spremnike --#
class SignalHandler:
  #-- Povezuje elemente grafickog
  # sucelja i sadrzi sve funkcije povratnog
  # poziva, kao i funkcije osvjezavanja grafickih
  # elemenata unutar sucelja --#
class Arduino:
  #-- Sadrzi funkcije za uspostavu serijske
  # komunikacije, slanje i primanje podataka --#
class TrafficLight:
  #-- Sadrzi sve podatke potrebne za simulaciju i upravljanje
  # sekvencijalnog rada raskrsca sa semaforima --#
class PLC:
  #-- Sadrzi sve podatke potrebne za simulaciju i upravljanje
  # osnovnog rada PLC-a --#
class ThermalChamber:
  #-- Sadrzi podatke potrebne za upravljanje i prikaz rada
  # toplinske komore edukacijske makete --#
```

Kod 13: Programska struktura grafičkog sučelja

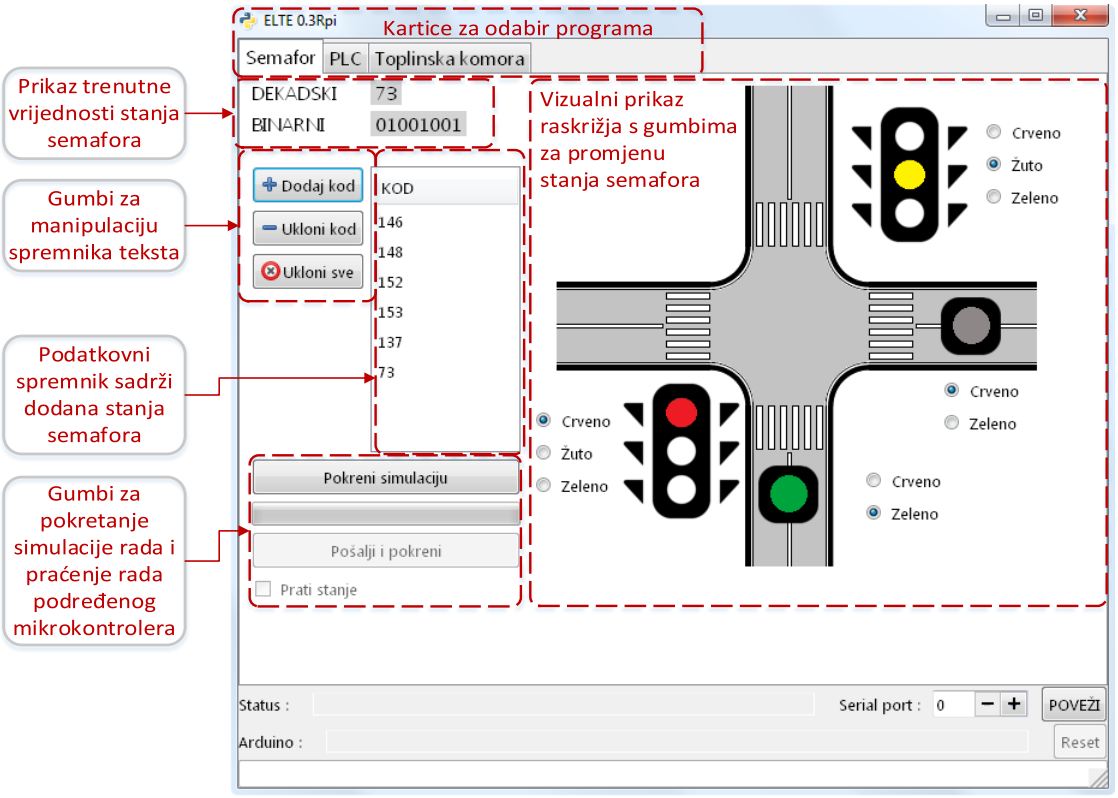


Slika 32: Zaglavlje grafičkog sučelja

Slika 32 prikazuje zaglavlje programa i njegove osnovne elemente. Zaglavlje programa služi za uspostavljanje serijske komunikacije s podređenim mikrokontrolerom. Nakon uspostave serijske komunikacije moguće je ručno unositi poruke za rad mikrokontrolera. Zaglavlje sadrži dvije obavijesne trake koje u svakom trenutku prikazuju u kojem stanju se nalaze nadređeni i podređeni kontroler.

5.5.1 Grafičko sučelje za sekvencijalno upravljanje semaforima

Slika 33 prikazuje osnovne elemente grafičkog sučelja za sekvencijalno upravljanje semaforima. Stanja semafora određuju se pritiskom na željeno stanje semafora uz odgovarajući semafor. Nakon svake promjene stanje pokazuje se binarna i dekadska vrijednost novog stanja. Pritiskom na gumb *Dodaj kod* trenutno stanje semafora dodaje se u podatkovni spremnik (engl. *data buffer*). Korisnik u svakom trenutku može simulirati rad semafora s postojećim stanjima spremljenima u podatkovnom spremniku pritiskom na gumb *Pokreni simulaciju*. Prilikom simulacije svjetla semafora i vrijednosti stanja odražavaju trenutni korak sekvencijalnog upravljanja. Pritiskom na tipku *Pošalji i pokreni*, stanja iz podatkovnog spremnika šalju se podređenom mikrokontroleru za upravljanje semaforima edukacijske makete. Odabirom opcije *Prati stanje* svjetla semafora i vrijednosti stanja semafora postavljaju se na trenutnu vrijednost sekvencijalnog upravljanja, odnosno stanje semafora unutar grafičkog sučelja odgovara stanju semafora na edukacijskoj maketi.

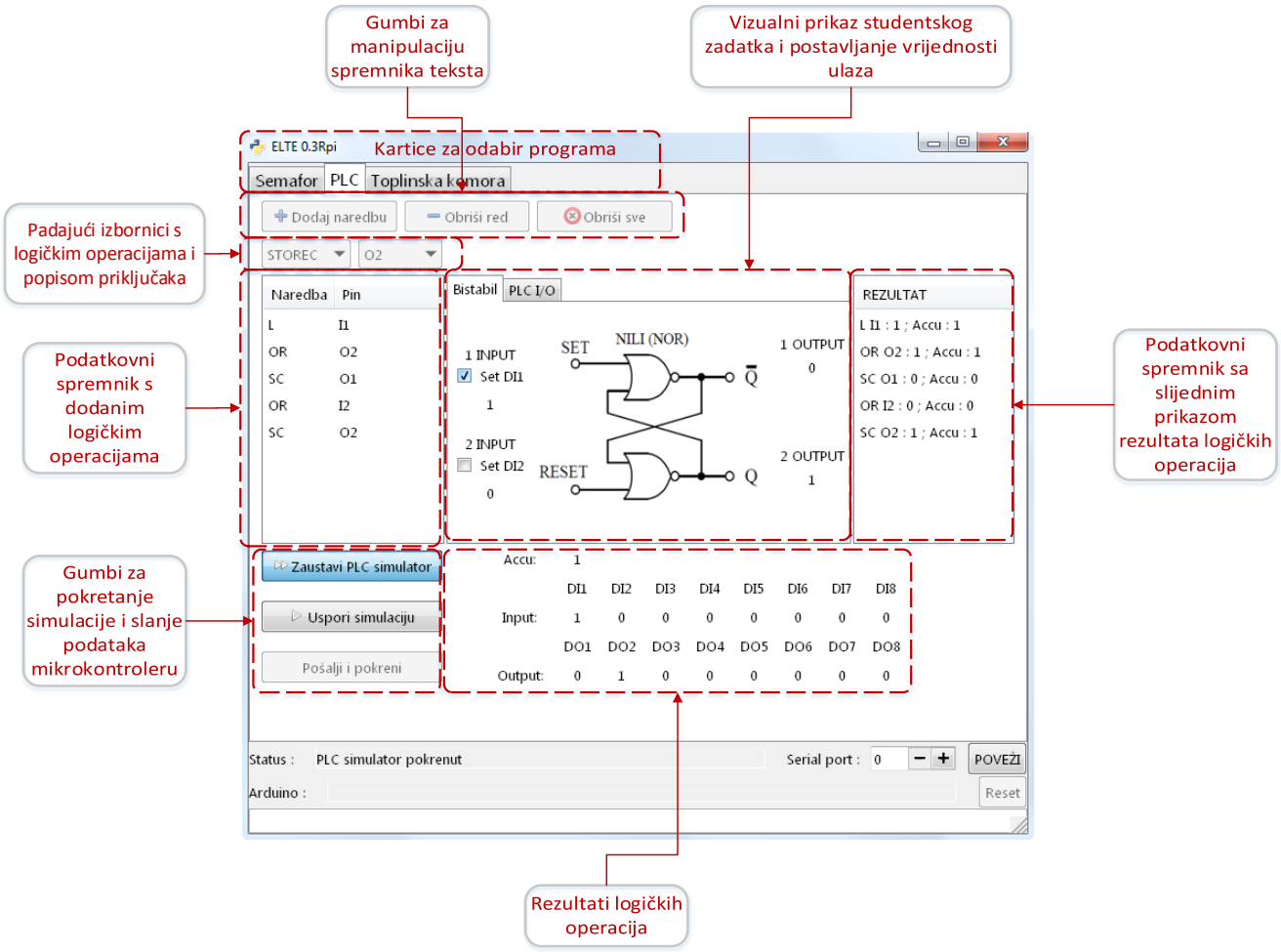


Slika 33: Grafičko sučelje za sekvencijalno upravljanje semaforima

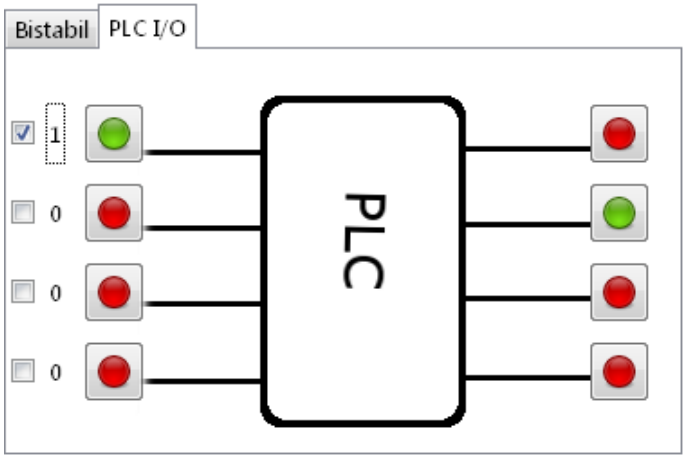
5.5.2 Sučelje za simulaciju osnovnog rada PLC-a

Slika 34 prikazuje osnovne elemente grafičkog sučelja za simulaciju osnovne funkcionalnosti PLC-a. Odabir logičke operacije i priključka na koji se logička operacija odnosi provodi se odabirom iz padajućeg izbornika. Pritiskom na tipku *Dodaj naredbu*, odabrana logička operacija i priključak dodaju se u podatkovni spremnik. Pritiskom na tipku *Pokreni PLC simulator* pokreće se simulacija osnovnog rada PLC-a s logičkim operacijama spremljenima u spremniku. Prilikom izvođenja simulacije rada PLC-a, korisnik može mijenjati stanja ulaznih priključaka čime utječe na rezultat logičke operacije. Rezultat logičkih operacija nakon svakog koraka prikazuje se u podatkovnom spremniku *Rezultat*. Pritiskom na tipku *Pošalji i pokreni*, vrijednosti iz podatkovnog spremnika šalju se preko serijske veze podređenom mikrokontroleru. Povratne informacije od podređenog mikrokontrolera nema, no moguće je izvršiti usporedbu rezultata postavljanjem ulaznih stanja fizičkih i virtualnih priključaka na iste vrijednosti. Vizualni prikaz bistabila služi za jednostavnije određivanje redoslijeda logičkih operacija za simuliranje rada bistabila. No unutar tog pregleda nije moguće postaviti vrijednosti ulaza za priključke **I3** i **I4**. Za postavljanje vrijednosti drugih ulaza i vizualni prikaz rezultata simulacije pogodnija je

forma u kartici *PLC I/O* prikazana na slici 35.



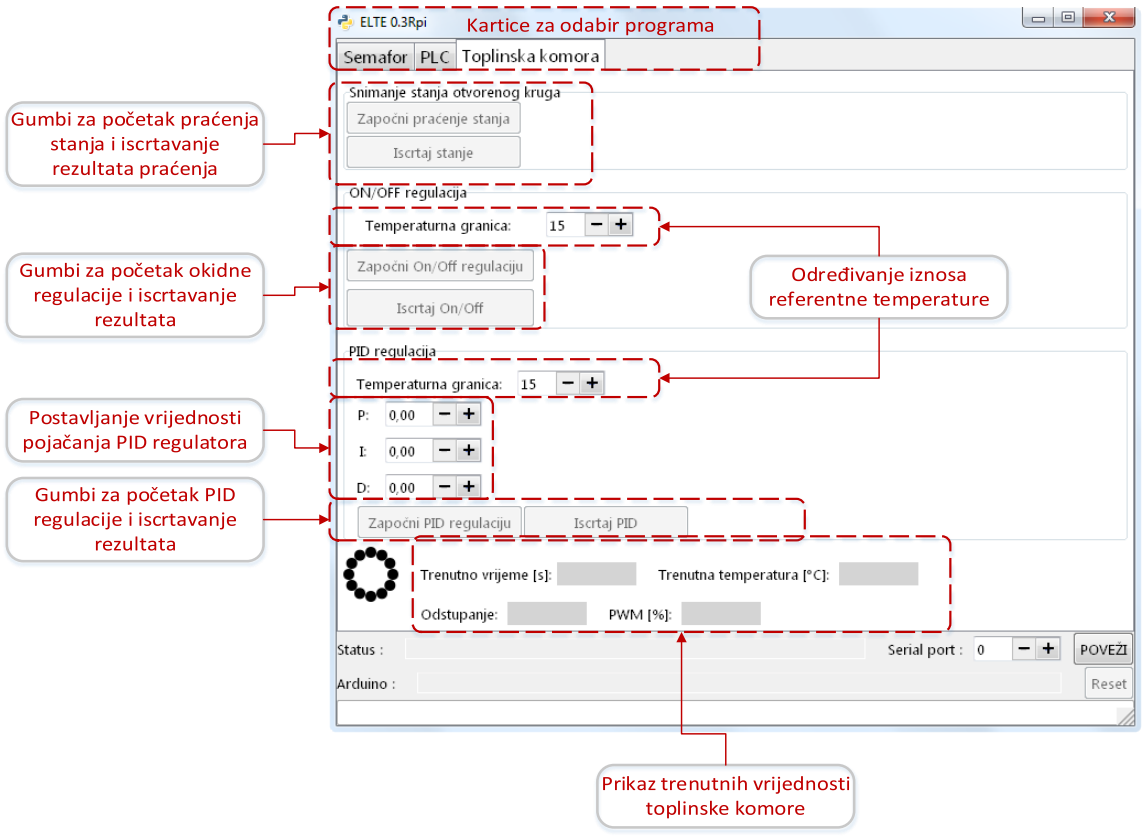
Slika 34: Grafičko sučelje za simulaciju osnovnog rada PLC-a



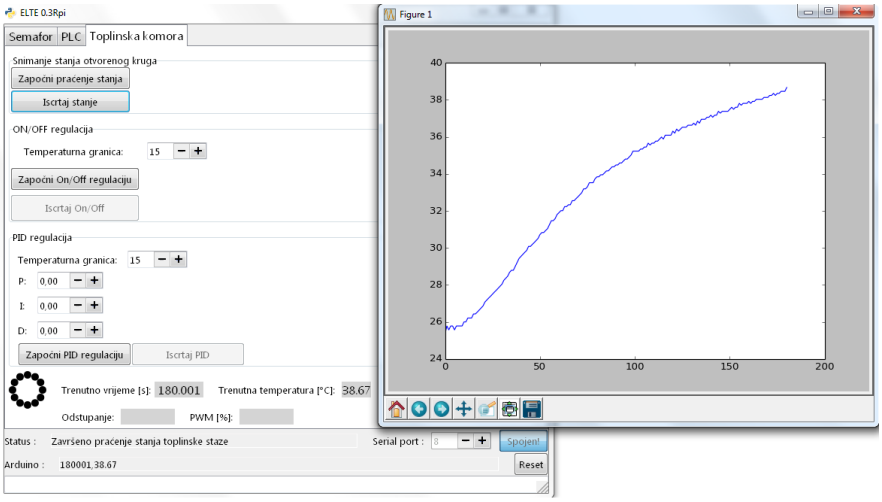
Slika 35: Vizualni prikaz rezultata simulacije osnovnog rada PLC-a za sve ulaze i izlaze

5.5.3 Sučelje za regulaciju temperature toplinske komore

Slika 36 prikazuje osnovne elemente grafičkog sučelja za praćenje stanja i regulaciju temperature toplinske komore. Zbog tehničkih ograničenja Raspberry Pi računala nije moguće iscrtavanje grafova u realnom vremenu unutar GTK+ sučelja. Pritiskom na gumb *Započni praćenje stanja*, podređenom mikrokontroleru šalje se poruka za praćenje stanja otvorenog regulacijskog kruga toplinske komore. Podatci o temperaturi se potom pohranjuju lokalno na Raspberry Pi računalu u obliku tekstualne datoteke. Spremanje podataka traje sve dok korisnik ne prekine praćenje stanja ponovnim pritiskom na gumb. Spremljeni podatci mogu se iscrtati pritiskom na gumb *Iscrtaj stanje* (slika 37). Graf se prikazuje u zasebnom prozoru i koristi vlastitu programsku petlju, stoga je prije nastavka rada u glavnom programu potrebno zatvoriti prozor grafa. Isti princip primjenjen je za praćenje stanja temperature zraka unutar toplinske komore za okidnu i PID regulaciju. Tijekom bilježenja podataka stanja toplinske komore, vrijednosti bitne za proces u trenutnom koraku uzorkovanja prikazuju se u donjem dijelu kartice.

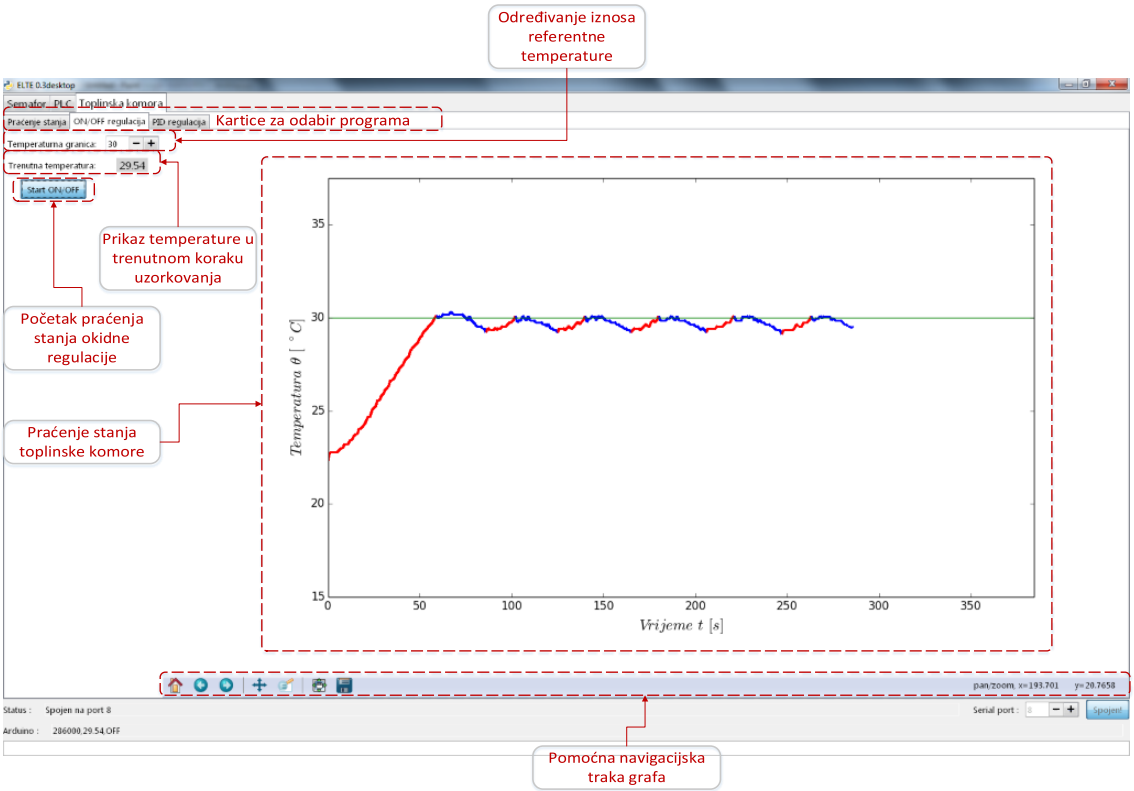


Slika 36: Grafičko sučelje za praćenje stanja i regulaciju temperature toplinske komore



Slika 37: Iscrtavanje prikupljenih podataka

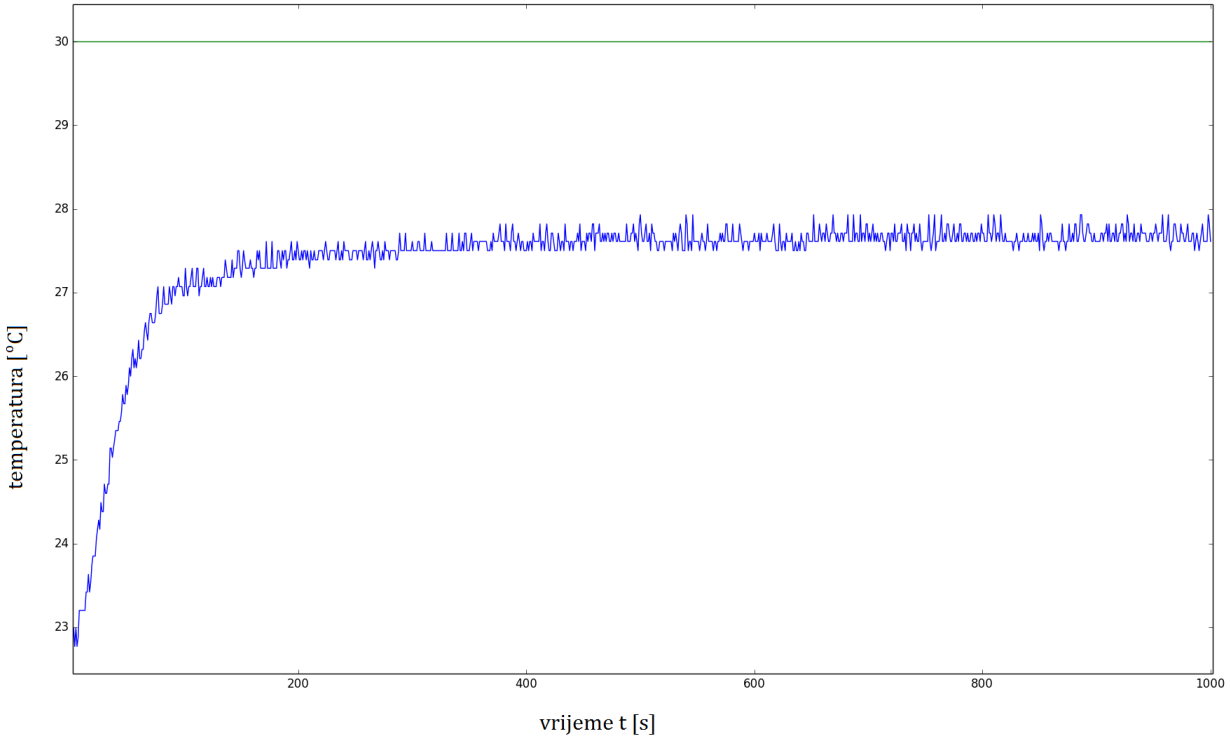
Za potrebe ovog diplomskog rada razvijeno je sučelje koje može prikazivati promjenu temperature zraka toplinske komore unutar grafičkog sučelja. Vrijednosti grafa ažuriraju se svakim korakom uzorkovanja i omogućavaju korisniku jasniji pregled stanja toplinske komore. Sučelje je podijeljeno u tri kartice unutar kojih je moguće zadati parametre regulacije toplinskog procesa za željeni oblik regulacije i započeti praćenje stanja. Slika 38 prikazuje osnovne elemente grafičkog sučelja za okidnu regulaciju temperature toplinske komore.



Slika 38: Grafičko sučelje za praćenje stanja okidne regulacije temperature toplinske komore

6 Usporedba relejne i PID regulacije temperature toplinske komore [12] [31]

Unutar grafičkog sučelja korisnik može proizvoljno odrediti iznose pojačanja K_p, K_i, Kd. Za dobivanje općih vrijednosti pojačanja koje osiguravaju zadovoljavajuću kvalitetu odziva provodi se podešavanje parametara primjenom Takahashijeve metode. Takahashijeva metoda je empirijska metoda određivanja pojačanja na temelju odziva otvorenog (nema djelovanja regulatora na upravljačku veličinu) ili zatvorenog regulacijskog kruga diskretnog PID regulatora. Provođenje eksperimenta u zatvorenom regulacijskom krugu sastoji se od povećavanja pojačanja K_p proporcionalnog djelovanja do iznosa koji uzrokuje pojavu neprigušenih oscilacija, tj. do granice stabilnosti sustava. Toplinski proces toplinske komore je dosta dobro opisan modelom dinamičkog aperiodskog člana prvog reda (P1 član), te ga nije moguće moguće dovesti na rub stabilnosti povećanjem pojačanja proporcionalnog djelovanja (slika 39).



Slika 39: Odziv toplinskog procesa pri čistom proporcionalnom djelovanju regulatora uz proporcionalno pojačanje $K_p = 100$

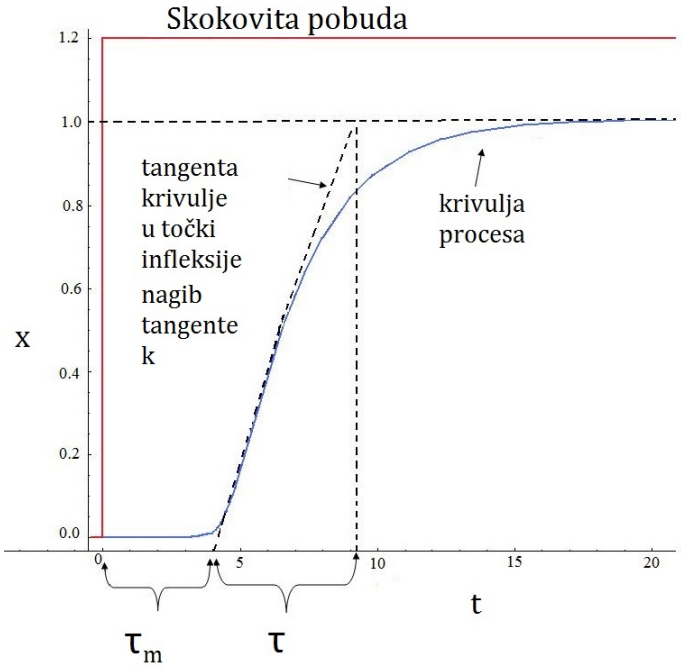
Zbog toga se parametri PID regulatora određuju na temelju odziva procesa na skokovitu pobudu u otvorenom regulacijskom krugu. Na krivulju procesa postavlja se tangenta u točki infleksije krivulje.

Potrebno je grafički očitati sljedeće parametre:

- τ_m - nadomjesno mrtvo vrijeme.
- k - nagib tangente

Tablica 11: Takahashijeva tablica za određivanje vrijednosti pojačanja [31]

Tip regulatora	K_p	T_0/T_i	T_d/T_0
P	$\frac{1}{k(\tau_m + T_0)}$	-	-
PI	$\frac{0,9}{k(\tau_m + T_0)} - \frac{0,135T_0}{k(\tau_m + 0,5T_0)^2}$	$\frac{0,27T_0}{K_p k(\tau_m + 0,5T_0)^2}$	-
PID	$\frac{1,2}{k(\tau_m + T_0)} - \frac{0,3T_0}{k(\tau_m + 0,5T_0)^2}$	$\frac{0,6T_0}{K_p k(\tau_m + 0,5T_0)^2}$	$\frac{0,5}{K_p k T_0}$



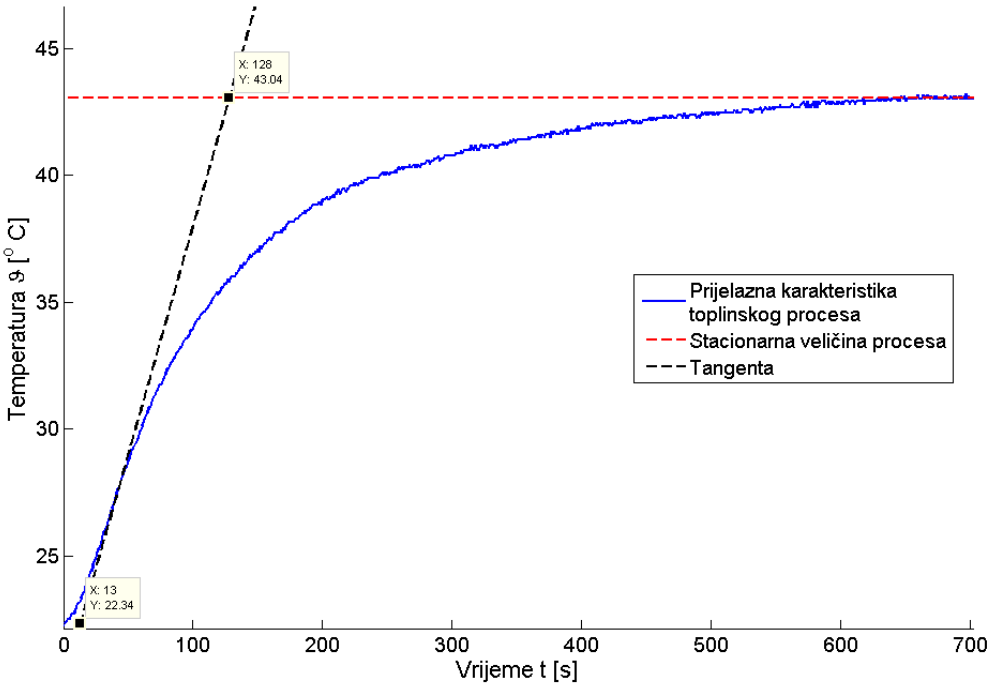
Slika 40: Određivanje vrijednosti potrebnih parametara za Takahashijevu metodu [32]

Slika 41 prikazuje odziv sustava na skokovitu pobudu. Parametri potrebni za proračun vrijednosti iznose:

$$T_0 = 1\ [s] \tag{17}$$

$$k = 0,18$$

$$\tau_m = 13\ [s]$$



Slika 41: Odziv sustava na skokovitu pobudu u otvorenom regulacijskom krugu s tangentom u točki infleksije

Vrijednosti pojačanja PID regulatora na temelju Takahashijeve metode (tablica 11) eksperimenta otvorenog regulacijskog kruga iznose:

$$K_p = \frac{1,2}{k(\tau_m + T_0)} - \frac{0,3T_0}{k(\tau_m + 0,5T_0)^2} \tag{18}$$

$$K_p = \frac{1,2}{0,18 \cdot (13+1)} - \frac{0,3}{0,18 \cdot (13+0,5)^2} = 0,467$$

$$\frac{T_0}{T_i} = \frac{0,6T_0}{K_p k(\tau_m + 0,5T_0)^2} \tag{19}$$

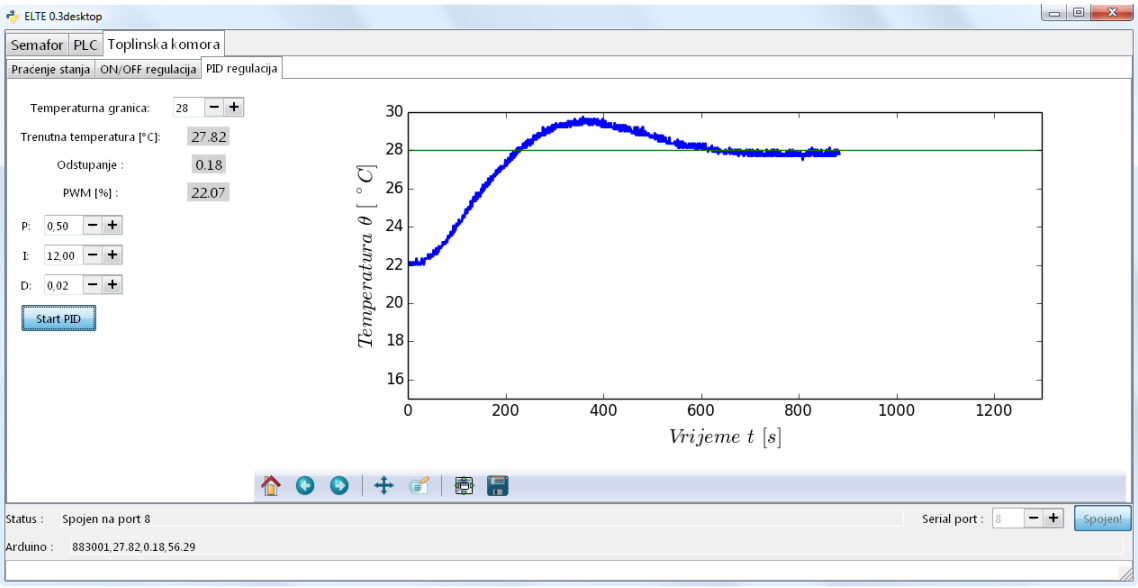
$$\frac{T_0}{T_i} = \frac{0,6}{0,467 \cdot 0,18 \cdot (13+0,5)^2} \approx 0,04$$

$$K_i = \frac{K_p}{T_i} T_0 = \frac{0,467}{0,04} = 11,675$$

$$\frac{T_d}{T_0} = \frac{0,6T_0}{K_p k(\tau_m + 0,5T_0)^2} \tag{20}$$

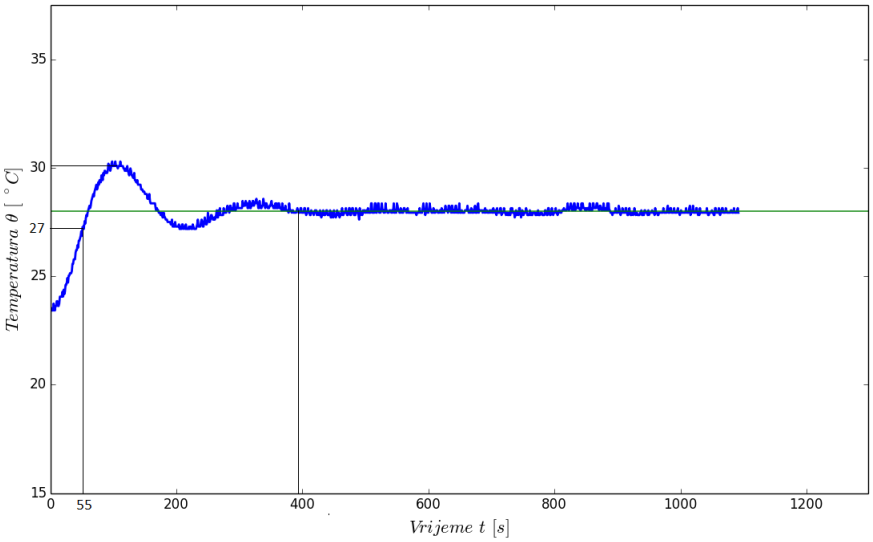
$$\frac{T_d}{T_0} = \frac{0,6}{0,467 \cdot 0,18(13+0,5)^2} \approx 0,033$$

$$K_d = \frac{K_p T_d}{T_0} = 0,467 \cdot 0,033 \approx 0,02$$



Slika 42: Odziv sustava na skokovitu pobudu s pojačanjima regulatora dobivenima primjenom Takahashijeve metode

Slika 42 prikazuje odziv sustava s vrijednostima pojačanja regulatora dobivenim primjenom Takahashijeve metode. Odziv zatvorenog regulacijskog kruga karakteriziran je razmjerno velikim iznosom nadvišenja (oko 25%) i velikim iznosom vremena smirivanja (oko 600 s, odnosno 10 minuta) uslijed malog iznosa proporcionalnog pojačanja, te kao takav ne udovoljava zahtjevima na kvalitetu odziva. Za dobivanje vrijednosti pojačanja regulatora koji zadovoljavaju zahtjeve za kvalitetom odziva parametri regulatora podešavaju se postupkom pokušaja i popravaka, a gdje se kao polazna točka koristi podešenje dobiveno Takahashievim postupkom.



Slika 43: Odziv sustava na skokovitu pobudu s proizvoljno odabranim pojačanjima regulatora

Slika 43 prikazuje odziv toplinske komore za sljedeće vrijednosti pojačanja regulatora:

$$K_p = 15$$

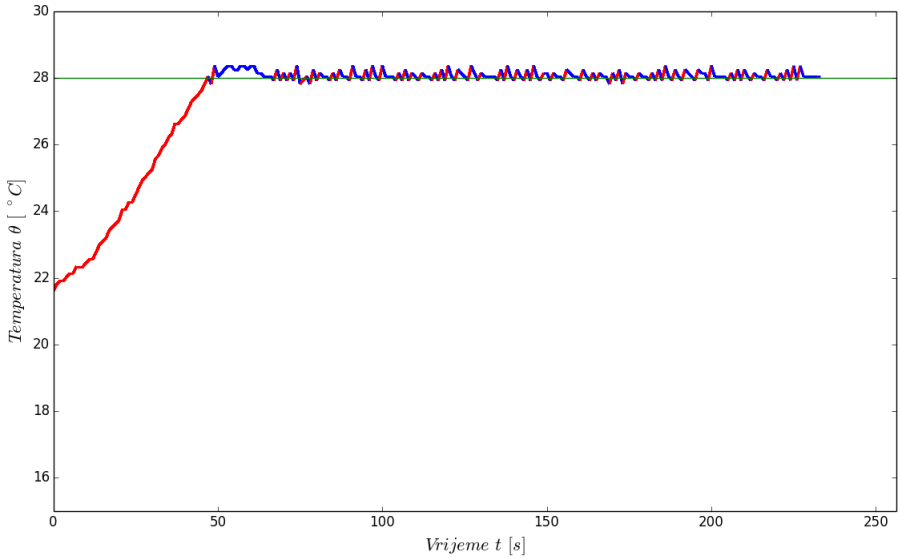
$$K_i = 1$$

$$K_d = 3$$

Tablica 12 sadrži vrijednosti parametara za određivanje kvalitete odziva sustava.

Tablica 12: Vrijednosti parametara za određivanje kvalitete odziva

	Iznos parametra	Komentar
T_r	$55\,s$	zadovoljavajuće vrijeme porasta funkcije, upravljačka veličina brzo ulazi u zasićenje
T_s	$385\,s$	prihvatljivo vrijeme smirivanja
M_p	7%	prihvatljiv prebačaj
e_0	0	regulator uklanja trajno regulacijsko odstupanje



Slika 44: Odziv sustava na skokovitu pobudu s pojačanjima regulatora dobivenim primjenom Takahashijeve metode

Slika 44 prikazuje okidnu regulaciju temperature toplinske komore. Usporedbom odziva primjetno je da je okidna regulacija bolji odabir za regulaciju temperature zraka unutar toplinske komore, ukoliko je relejno djelovanje prihvatljivo sa stanovišta rada izvršnog člana (u ovom slučaju grijaćeg tijela odnosno žaruljice).

7 Zaključak

U diplomskom radu prikazan je postupak projektiranja i razvoja programa za upravljanje objektima edukacijske makete. Opisani su elementi makete i njihov princip rada. Za upravljanje elementima makete osmišljen je upravljački program koji se nalazi na Arduino Uno mikrokontroleru. Upravljački program osim upravljanja različitim procesima (objektima upravljanja) ima zadaću praćenja stanja trenutnog objekta upravljanja i slanje tih podataka nadređenom kontroleru, Raspberry Pi računalu. Za promjenu upravljačkog programa mikrokontroler mora zaprimiti odgovarajuću poruku od strane nadređenog kontrolera, radi čega je precizno definiran komunikacijski protokol, odnosno format i sadržaj svake poruke. Mikrokontroler mijenja upravljački program samo ako dobije poruku koja ispunjava sve uvjete određene protokolom.
U drugom dijelu rada prikazama je izrada korisničkog sučelja na nadređenom kontroleru. Za programski jezik odabran je Python, a za izradu grafičkog sučelja korišten je GTK+ programski razvojni paket. Prilikom izrade grafičkog sučelja vodilo se računa da sučelje bude jasno, pregledno, funkcionalno, te da je kompatibilno s različitim operacijskim sustavima. Korisničko sučelje sastoji se od simulacijskog i upravljačkog dijela. Simulacijski dio programa omogućuje korisniku da odredi parametre objekta upravljanja, te da uz zadane parametre simulira rad realnog objekta. Upravljački dio programa sastoji se od proslijeđivanja parametara upravljanja podređenom mikrokontroleru i praćenja stanja upravljanog objekta regulacije. Za praćenje stanja toplinske komore razvijena su dva različita sučelja: jedno optimirano za radna Raspberry Pi računalu, a drugo za rad na *desktop* računalu.
Daljnje unaprijeđenje projekta ovisit će ponajviše o povratnim informacijama njegovih korisnika. Hardverske promjene uključuju zamjenu Arduino Uno mikrokontrolera s Arduino Mega mikrokontrolerom, ili zamjenu Raspberry Pi Model B računala s Raspberry Pi 2 Model B računalom.
Sva razvojna rješenja koja su razmatrana u radu dana su u prilogu.

8 Popis literature i izvora

Literatura

[1] Essert, Grilec, Skalicki, Zorc, Deur, Pavković : **Vježbe iz elektrotehnike, elektronike i električnih strojeva**, Fakultet strojarstva i brodogradnje, Zagreb, 2008.

[2] Arduino :`http://www.arduino.cc/`, zadnji pristup 15.3.2015.

[3] Raspberry Pi službena stranica: `http://www.raspberrypi.org/`, zadnji pristup 15.3.2015.

[4] Watt-Boulton centrifugalni regulator : `http://upload.wikimedia.org/wikipedia/commons/b/b7/Centrifugal_governor.svg`, zadnji pristup 15.3.2015.

[5] Arduino - Uno 3: `http://www.arduino.cc/en/Main/arduinoBoardUno`, zadnji pristup 15.3.2015.

[6] Atmel ATmega328 datasheet: `http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf`, zadnji pristup 15.3.2015.

[7] Raspberry Pi službena stranica: `http://www.raspberrypi.org/products/model-b/`, zadnji pristup 15.3.2015.

[8] Wikipedia : `http://en.wikipedia.org/wiki/Raspberry_Pi#Specifications`, zadnji pristup 15.3.2015.

[9] Raspbian : `http://www.raspbian.org/`, zadnji pristup 15.3.2013.

[10] Predavanja iz kolegija 'Digitalna logika' : `http://www.fer.unizg.hr/_download/repository/DL11_08-09_Stand._sekv1._skl.pdf`, FER, Sveučilište u Zagrebu, 2008./2009., zadnji pristup 15.3.2015.

[11] Posmačni registar 74HC595 *datasheet* : **Texas Instruments** - `http://www.ti.com/lit/ds/scls041h/scls041h.pdf`, zadnji pristup 15.3.2015.

[12] Dubravko Majetić : **Skripte za predavanja iz kolegija : 'Upravljanje i regulacija'**, FSB, Zagreb, 2012.

[13] Temperaturni senzor LM35DZ *datasheet*: **Texas Instruments** - http://www.ti.com/lit/ds/symlink/lm35.pdf, zadnji pristup 15.3.2015.

[14] Šurina, T. : **Automatska regulacija**, Školska knjiga, Zagreb, 1990.

[15] Isermann, R. : **Digital Control Systems, Vol. 1, Chap. 5**, Springer-Verlag, Berlin, 1989.

[16] Arduino - vrsta podataka :http://arduino.cc/en/Reference/Double, zadnji pristup 15.3.2015.

[17] Arduino - TimerObject : http://playground.arduino.cc/Code/ArduinoTimerObject

[18] Arduino - Boolean : http://arduino.cc/en/Reference/Boolean, zadnji pristup 16.3.2015.

[19] Python : https://www.python.org/

[20] Raspbian Repository : http://www.raspbian.org/RaspbianRepository

[21] Essert, M. : **Python - digitalni udžbenik**, Odjel za matematiku, Sveučilište Josipa Jurja Strossmayera, Osijek, 2007.

[22] Zelle, J. : **Python Programming: An Introductory to Computer Science**, Franklin, Beedle & Assoc., Oregon, 2004.

[23] GTK+ : http://www.gtk.org/

[24] GTK+ Python API : http://lazka.github.io/pgi-docs/Gtk-3.0/index.html

[25] NumPy : http://www.numpy.org/

[26] matplotlib : http://matplotlib.org/

[27] matplotlib API : http://matplotlib.org/api/

[28] Tosi S. : **Matplotlib for Python Developers**, Packt Publishing, Birmingham, 2009.

[29] pySerial : http://www.gtk.org/

[30] pySerial API: http://pyserial.sourceforge.net/pyserial_api.html

[31] Perić, N. - **Materijali za studente : PID regulator**: http://act.rasip.fer.hr/materijali/11/pid_Control_15_1_2007.pdf, FER, Sveučilište u Zagrebu, 2007.

[32] Podešavanje parametara PID regulatora : https://controls.engin.umich.edu/wiki/index.php/PIDTuningClassical, zadnji pristup 16.3.2015.

Prilog

1. DVD koji sadrži :

    - Presliku (engl. *image*) Raspbian operacijskog sustava
    - Arduino kod podređenog mikrokontrolera
    - Kod grafičkog sučelja za Raspberry Pi
    - Kod grafičkog sučelja za *desktop* verziju
    - Proširenu električnu shemu za spajanje edukacijske makete s Arduino Uno mikrokontrolerom
    - Kod za digitalnu obradu (filtriranje) temperature unutar mikrokontrolera